



USER'S MANUAL

version 16.6

No Magic, Inc.
2009

All material contained herein is considered proprietary information owned by No Magic, Inc. and is not to be shared, copied, or reproduced by any means. All information copyright 1998-2009 by No Magic, Inc. All Rights Reserved

CONTENTS

Contents 3

1	INTRODUCING MAGICDRAW	22
	About MagicDraw and UML	22
	MagicDraw Editions and Features	23
	MagicDraw Editions	23
	MagicDraw Personal Edition	23
	MagicDraw Standard Edition	24
	MagicDraw Architect Edition	24
	MagicDraw Professional Edition	24
	MagicDraw Enterprise Edition	25
	MagicDraw Community Edition	25
	MagicDraw Reader Edition	25
	MagicDraw Features	26
	Reports Generation	26
	Floating License Server	26
	Teamwork License Server	26
	Code and Database Engineering	27
	OpenAPI	27
	Integrations	27
	MagicDraw Customization	28
	MagicDraw Plugins	28
	MagicDraw RConverter	29
	MagicDraw Welcome Screen	30
	MagicDraw News Reader	35
	MagicDraw Documentation and Support	37
	New and Noteworthy	37
	Manuals and User Guides	37
	Help	38
	Hints Associated to User's Actions	38
	Productivity Tips Displayed in Progress Window	39

CONTENTS

Tutorials	40
Other Documentation	42
Support	43
FAQ	43
Newsgroups	43
E-mail	43
Bug Reports	43
View and submit internal errors	47

2 GETTING STARTED 48

System requirements	48
Java Virtual Machine (JVM)	49
Operating System - dependent issues	50
Installation Procedure	50
Windows 2000/2003/NT/XP/Vista*	50
Unix	51
MAC OS X	51
All other platforms instructions (no install version)	51
Licensing Information Display	52
MagicDraw Configuration Files Location	53
User Registration	54
Registration Workflow	56
1. Add a License	57
2. Registering	58
3. Confirming Your Registration	60
4. Logging in to Your Dedicated Area at www.magicdraw.com	60
Bug Report	62
Used Licenses Management	62
Illegal Usa	64
Notifications of Registration	65
Troubleshooting	66
Updating	67
Auto-Check for Updates dialog box	68

3 USING MAGICDRAW 69

Perspective Selection and Customization 69

Customizing MagicDraw Perspectives 70

MagicDraw Startup dialog box 72

Select Perspective dialog box 73

Customize Perspectives dialog box 76

MagicDraw User Interface 78

Menu System 80

File menu 80

Edit menu 83

View menu 87

Layout menu 89

Diagrams menu 92

Options menu 93

Tools menu 93

Analyze menu 97

Teamwork menu 99

Window menu 99

Help menu 101

Toolbars 102

Customizing toolbars 102

Main Toolbar 105

External Tools Toolbar 105

Using the Browser 105

The Browser window parts 106

Containment tree 109

Data branch 114

Code engineering sets 115

Diagrams tree 118

Inheritance tree 120

Model Extensions Tree 121

Search Results Tree 123

Working with Model Elements in the Browser Tree 124

Multiple selection 125

Documentation/Zoom Control/Properties 126

Documentation tab 126

Zoom Control tab 127

Properties tab 128

Setting Environment Options 129

General pane 131

Diagram pane 135

Browser pane 139

Teamwork pane 141

Floating pane 142

CVS pane 143

Update pane 145

Network pane 146

Keyboard pane 147

Plugins pane 149

Resources pane 150

Path Variables Pane 152

Launchers Pane 153

Experience Pane 154

External Tools 156

Scripts 158

EMF UML2 (v2.x) XMI Options Pane 159

EMF UML2 (v1.x) XMI Options Pane 161

Look and Feel: Controlling the Interface 163

Single and Multiple Windows interface styles 164

Assigning Shortcut Keys 165

4 WORKING WITH PROJECTS 166

Creating a Project 166

Creating a new project 166

Working with multiple projects 167

Creating a new project from the existing source code 169

Creating a new project from a previously created template 170

Creating a new Use Case project 171

Saving a Project 173

Autosave 174

Opening a Project 175

Importing a Project 176

Exporting a Project 176

- Exporting a module of a project 176
- Exporting a project as a template 176
- Exporting a project as MOF 177
 - MOF Domain model 178
 - MOF export 178
 - Metamodeling template 180
 - Validation constraints 180
- Exporting a project as an EMF UML2 (v2.x) XMI 181
- Exporting a project as an EMF UML2 (v1.x) XMI 181
- Exporting to BPEL 182

Setting Project Options 186

- Project Options 186

Searching 189

- Java Regular Expressions 195
 - Metacharacters 195
 - Character Sets 195
 - Grouping 196
 - Quantifiers 196
 - Boundary Matchers 197
 - Embedded Flag Expressions 197

- Replacing 198

Project Partitioning 200

- Partitioning the model 200
- Exporting the module of a project 201
- Sharing the module of a project 203
- Managing modules (the Modules dialog box) 205
- Dependencies Between Elements 206
 - Package dependencies by relationship 207
 - Dependencies by reference 210
 - Diagram dependencies 210
- The Package Dependencies window 212
- Unresolved dependencies 218
- Using the module of a project 218
 - The Use Module Wizard 219
 - Reusing model parts between models 222
- Reloading the module of a project 224

- Importing the module of a project **224**
- Working with partially loaded projects **225**
- Advanced Concepts **226**
 - Indexing 226
 - Missing elements for the proxies (orphan proxies) 228

Ecore Support **231**

- Introduction **231**
- Preparing Ecore Models **231**
 - Annotation modeling 237
- Exporting **240**
- Validation **242**
- Importing **243**

Working with Standard Profiles **243**

- Standard Profiles as System Resources **243**
 - Plugin and Profile Versions 243
- Standard Profiles in Teamwork Server **244**

5 WORKING WITH DIAGRAMS **245**

Working with Diagrams **246**

- Diagram Basics **246**
- Diagrams Dialog Box **249**
- Diagram Properties **250**
- Diagram name and its context name synchronization **253**

Diagram Frame **254**

Table with diagram information **256**

Drawing Shapes **257**

Drawing the Shapes of the Diagrams **262**

Drawing Relationship Paths **263**

Inserting a Shape on the Path **266**

Creating Relations from the Model **268**

Smart Manipulation **269**

Selection and Multiple Selections **272**

Copying/Pasting Text or Images to Diagrams **275**

Nesting Image Shapes **276**

Dragging, Copying, Cutting, and Pasting **277**

CONTENTS

Changing the diagram type **287**

Zooming **290**

Using the Grid **291**

Layout **292**

- Orthogonal Layout Tool **293**

- Hierarchic Layout Tool **294**

- Tree Layout Tool **296**

- Organic Layout Tool **299**

- Circular Layout Tool **300**

- Orthogonal Path Router **302**

- Organic Path Router **303**

- Class Diagram Layout Tool **303**

- Activity Diagram Layout Tool **305**

- Business Process Diagram Layout tool **307**

- Quick Diagram Layout feature **309**

- Label layout in the diagram **309**

 - Default label positions 309

 - Labels positions after moving a path, shape or related element 309

Showing Diagrams in Full Screen **312**

Floating Diagram Window **314**

Saving as an Image **314**

- Saving a diagram and selected symbols as an image **314**

- Setting image saving options **317**

Printing **318**

- Print Range tab 319

- Print Options Tab 321

- Print Header/Footer Tab 323

6 WORKING WITH MODEL ELEMENTS **325**

Specifying Model Elements **325**

- Specification dialog boxes **325**

 - General tab 326

 - Documentation/Hyperlinks tab 327

 - Attributes tab 329

 - Usage in Diagrams tab 330

 - Operations tab 331

CONTENTS

Template Parameters tab	332
Relations tab	334
Tags tab	335
Constraints tab	338
Buttons available in the Specification dialog boxes	339
Default Property Values	339
Sharing the default property values	341
Formatting Symbols	342
To set the symbol properties (fill color, text color, and others)	342
Changing symbol properties dialog for multiple symbols	342
To show/hide model element constraints, stereotype and/or tagged values on the diagram pane	343
Displaying icon or image on the symbol or instead the symbol	344
Style Engine	346
Symbol Property Styles Tree	346
Working with Properties Styles	353
Properties extension by diagram	355
Properties Inheritance	358
General Style Properties	358
Shape, Path and Diagram Properties	358
Stereotype properties	360
Defining Hyperlinks Between Elements	362
Adding a hyperlink to the model element	363
Edit Hyperlink dialog box	367
Owner of the model element	369
Owner display mode	370
Qualified name starting from model library	371
Relations Changes Ownership when Client or Supplier is Moved to Other	
Owner	372
Converting An Element	373
Replacing An Element	373
HTML Editor	375
Copying/Opening Element URLs	385
 7 TOOLS	 387
Model Merge	388

Definitions	388
Introduction to Merging	388
3-way merge	389
2-way merge	390
Model Merge Concepts	391
Introductory Case Studies	391
Case Study 1 - 3-way Merge and Analysis	392
Case Study 2 - merging in Teamwork System	398
Case study 3 - Copying changes from branch to Trunk (in Teamwork) with conflict resolution	408
Case Study 4 - Representing DSL elements in the Merged results	421
Merging concepts in details	423
Change types	424
Accepting or Rejecting changes	425
Dependent changes	425
Conflicting changes	426
Building change tree	427
Analyzing Merging Results	429
The Merge window description	429
Viewing changes in diagrams	454
Finishing projects merge	462
Controlling Merge memory usage	465
Pattern Wizard	466
Creating Setters / Getters	475
Implementing or Overriding Operations	478
Model Transformation Wizard	480
Types of transformations	480
Model Transformation Wizard	481
Model Transformation Mapping	485
Transformation Mapping Options	486
Model Transformation Update	490
Profile Migration Transformation	491
Profile Migration Transformation mapping	491
Starting Profile Migration Transformation	493
Sample of the Profile Migration Transformation	493
Resource Manager	497
Spelling Checker	501

Spell checking as you type	501
Spell checking for the whole project or the selected scope	505
Checking spelling for the whole project	505
Checking spelling for the selected scope	506
Analyzing the Check Spelling (the Validation Results window)	507
Solving the spell checking errors	508
Setting the spell checking options	511
Spell checking options	512
Spell checking dictionaries	513
Defining properties of the customized element to be spell checked	514
Import Data to MagicDraw	515
Import data from Rational Software Architect/Modeler using MagicDraw	
RSXConverter	515
Import data from Rational Rose using MagicDraw RConverter	515
Import data from other tools	515
8 MODEL ANALYSIS	517
Model Visualizer	518
Class Diagram Wizard	519
Package Dependency Wizard	524
Displaying package dependencies	527
Package Overview Diagram Wizard	528
Hierarchy Diagram Wizard	533
Realization Diagram Wizard	537
Activity Decomposition Hierarchy Wizard	541
Content Diagram Wizard	546
Sequence Diagram from Java Source Wizard	549
Displaying related elements	555
Dependencies analysis	557
Usages Functionality	557
Dependent Elements functionality	557
Usages/Dependencies Search Options dialog box	558
Elements Using/Dependencies windows	561
Model Differencing	564
Models comparison	564
Understanding model differences	565

Diagrams Comparison	569
Metrics	572
Metric Suites	573
Displaying Metrics	573
Starting Metrics	574
Metrics window	577
Exporting Metrics	578
Comparing metrics	580
Metrics Options	580
Metrics Properties	583
Dependency Matrix	585
Creating the Dependency Matrix	586
Dependency Matrix View	587
Dependency Matrix pane	590
Working with a Dependency Matrix Template	593
Dialog boxes in Dependency Matrix functionality	595
Dependency Matrix Templates dialog box	595
The Add / Remove Elements dialog box	598
The Dependency Matrix Specification dialog box, tags group	599
Validation	601
Introduction	601
Constraint Types	601
Predefined Validation Suites	602
Validating	603
Validation Results Window	606
Validation Rules	609
OCL Constraints	614
Binary Constraints	615
Validation Suites	615
Advanced Topics	623
Global validation rules	623
Expressions in error messages	625
Modeling other types OCL2.0 constraints/expressions	626
Unsupported OCL2.0 features	628
Adding/customizing severity levels	628
Performance Issues	629

Active Validation **629**

- Detecting errors in the model **630**

 - Failure indicator 630

 - Marking errors in the Browser 632

 - Highlighting errors on the diagram 633

- Handling incorrect model **634**

- Changing the Active Validation Options **635**

- The Active Validation Suites **636**

 - Validating Parameters and Arguments Synchronization 636

 - Shape Ownership 637

 - UML model correctness 641

 - Validating the Orphaned Proxies (OP) 641

- Customizing the Active Validation **642**

- Validate element that has no representation in diagram **642**

Usage in Diagrams **644**

 - Searching for symbol usage in diagrams from the element specification dialog box 645

 - Searching for symbol usage in diagrams from the element shortcut menu 646

9 UML DIAGRAMS **647**

Architectural Views **648**

- Use Case View **648**

- Structural View **649**

- Behavioral View **649**

- Implementation view **649**

- Environment view **649**

Class Diagram **650**

- Class diagram elements **651**

Use Case Diagram **659**

- Use Case diagram elements **660**

Communication Diagram **662**

- Communication Diagram elements **663**

Sequence Diagram **665**

- Sequence diagram elements **666**

- Sequence diagram improvements **671**

 - State Invariant 671

 - Lost / Found Messages 673

CONTENTS

State Machine Diagram 674

State Machine Diagram elements 675

Protocol State Machine Diagram 680

Protocol State Machine Diagram elements 681

Activity Diagram 685

Activity Diagram elements 687

Smart Activity Diagram layout 696

Dynamic centerlines 696

Diagram orientation 697

Implementation Diagram 698

Component diagram overview 698

Deployment diagram overview 699

Implementation Diagram elements 700

Composite Structure Diagram 707

Composite Structure Diagram elements 708

Displaying existing Parts on the Composite Structure diagram creation 710

Interaction Overview Diagram 712

10 EXTENSION DIAGRAMS 715

General Information 715

Common Elements 717

User Interface Modeling Diagram 719

Overview 719

Why Prototyping User Interface Modeling? 719

Working with User Interface Modeling Diagrams 720

Starting Working with a User Interface Modeling Diagram 720

Creating a User Interface Model 721

Loading a Sample with an Already Created User Interface Model 721

User Interface Modeling 721

User Interface Modeling Components 722

Modifying a Table 729

Modifying a Tree 733

Nesting 733

Reusability 734

Specifying Elements 734

Icon Usage 737

CONTENTS

Using Symbol Properties	738
User Interface Prototyping	739
Case Studies for User Interface Modeling	740
Case Study 1 - Modelling User Interface for the Report Wizard Window	741
Case Study 2 - Slider Example	746
Case Study 3 - User Interface Prototyping Example	749
Content Diagram	752
Content Diagram Elements	753
Robustness Diagram	757
Robustness Diagram Elements	758
Web Diagram	759
Web Diagram Elements	760
CORBA IDL Diagram	763
CORBA IDL Diagram Elements	764
Generic DDL Diagram	765
Generic DDL Diagram Elements	766
Oracle DDL Diagram	766
Oracle DDL Diagram Elements	768
WSDL Diagram	769
WSDL Diagram Elements	770
XML Schema Diagram	771
XML Schema Diagram Elements	772
Time Diagram	775
Time Diagram Elements	776
Struts Diagram	776
Struts Diagram Elements	777
Networking Diagram	779
Networking Diagram Elements	780
Business Process Diagram	783
Business Process Diagram Elements	784
 11 MODEL ELEMENTS	 796
Common Model Elements in the Diagrams	796
Note, Comment	799
Anchor	800

CONTENTS

Constraint path	800
Image Shape	800
Separator	801
Documentation	801
UML Extension Elements	802
Stereotype	803
Stereotype Specification dialog box	806
Assigning a stereotype	808
Changing the stereotype display mode	809
Stereotype notation	811
Tags	811
Editing Tagged Value	813
Constraint	815
The Time and Duration Constraints	816
Working with Constraints	816
The Constraint Specification dialog box	818
OCL	819
Profiles	822
Profiles	823
Working with profiles	824
Creating Profiles	824
Using and Importing Profiles	824
Exporting Profiles	825
Action	825
Accept Event Action	826
Call Behavior Action	828
Call Operation Action	829
Opaque Action	831
Send Signal Action	831
Working with actions	832
Advancing actions: applying duration constraint	833
Actor	835
Association	837
Association End	841
Advancing actions: navigable owned association ends	847
Association in Use Case Diagrams	848
Attribute	849

CONTENTS

Class 854

- Working with classes **855**
- Creating A Structured Class **857**
- Design Patterns **859**
- Class presentation options **859**

Collaboration 862

Combined Fragment 865

Component 866

Connector 870

Containment 874

Data type 874

- Enumeration **876**
- Primitive **879**

Decision Node 879

Dependency 879

- Template Binding dependency **882**
- Abstraction **883**
- Usage **883**
- Package Merge **883**
- Package Import **884**
- Element Import **884**
- Access **884**
- Deployment **885**

Deployment Specification 885

Exception Handler 886

Extend 887

Flow Final Node 889

Fragment 890

- Alternative Fragment **890**
- Combined Fragment **891**

Function Behavior 893

Gate 894

The formal gate and actual gate usage in the sequence diagram **897**

Generalization 899

- Generalizable elements **900**

Generalization sets	901
Include	906
Information Flow	908
Information Item	909
Instance	910
Instance Specification	915
Interaction Use	916
Interface	917
Provided and Required Interfaces	918
Provided and Required Interfaces in the Composite Structure diagram	920
Provided/required interfaces in the Component diagram	922
Internal transition	922
Lifeline	923
Lifeline in the Sequence Diagram	925
Link	926
Manifestations	926
Message	927
Assigning/Creating operation for message	932
Assigning/Creating signal reception for message	933
Predecessors and activators	934
Message in Sequence Diagram	935
Creating nested activation	935
Model	938
Node	939
Structured activity node	942
Activity parameter node	943
Expansion Region and Expansion Nodes	943
If, Loop and Sequence Conditional Nodes	943
Object	944
Object Flow	945
Object Node	945
Opaque Behavior	947
Operation	948
Package	955
Working with packages	956
Parameter	960

Parameters synchronization with Arguments **960**

Rules of synchronization between parameters and arguments 962

Synchronization between Operation parameters and Behavior parameters 963

Synchronization between Activity parameters and Activity Parameter Nodes 967

Synchronization between Operation parameters and pins on Call Operation Action 969

Synchronization between Behavior parameters and pins of Call Behavior Action 970

Synchronization between Interaction parameters and Interaction Use arguments 973

Synchronization between Operation parameters and Message arguments 973

Synchronization between Interaction Parameters and Lifelines 976

The Parameters Synchronization dialog box 979

Port **987**

Pseudo State **991**

Initial **992**

Final state **992**

Terminate **992**

Entry Point **992**

Exit Point **992**

Deep History **993**

Shallow History **993**

Junction **993**

Choice **993**

Fork/Join **994**

Realization **994**

Reception **997**

Send Signal Action **999**

State **1001**

Changing State to Composite/submachine/orthogonal State **1005**

Composite State **1006**

Submachine **1007**

Adding connection point reference **1007**

Defining State Invariant **1008**

Assigning behavior to state **1008**

Subsystem **1009**

Swimlane **1010**

Template / Parameterized class **1015**

Transition **1017**

CONTENTS

Event types	1019
Enhanced Event assignment to transition	1020
Use Case	1020
Use Case Extension	1023
Value Specification	1024
 APPENDIX: LIST OF ICONS	 1026
Icons from the MagicDraw GUI	1026
Main Toolbar	1026
Buttons from the diagram list toolbar	1029
Diagram main toolbar	1030
Buttons from the Browser window	1033
Floating, Auto-hide and Close buttons	1034
Buttons from the Containment tree	1035
Buttons from the Inheritance tree	1036
Buttons from the Diagrams tree	1037
Buttons from the Model Extensions tree	1037
Buttons from the Search Results tree	1038
Diagram Icons	1039
Icons of general elements	1041
Icons of relationships	1055
Icons from Modules and Profile mechanism	1061
Index	1062

1 INTRODUCING MAGICDRAW

About MagicDraw and UML

Today's graphical software can be extremely complex in its structure and architecture, but that does not mean it must be difficult to use. We have learned much from the hardware industry, where everything you see is scattered pieces. This approach also works well in the software world – objects at a higher abstraction level are treated like “software pieces.” To simplify the process further, we may use pictures instead of textual descriptions to show the relationships between objects in a complex system. Though pictures work better than textual descriptions alone, experience has proven that communicating complex ideas effectively requires more than simple flowcharts.

Early methodologies, such as Booch notation, OMT, and others served the same purpose: to graphically express the software's architecture information. However, these methodologies accomplished this in slightly different ways and with different levels of thoroughness. In 1994, Grady Booch, Jim Raumbaugh, and Ivar Jacobson came together to unify their varied methods and experience. The UML (Unified Modeling Language) was the fruit of their joint effort. UML was crafted with two objectives: To reflect the best practices of the industry and to demystify the process of software system modeling.

In short, UML provides standardized pictures of your software applications and allows your development team to quickly grasp the functionality contained within the application. UML is a language and a process with neutral notation. This means that you can use it to design your entire OO system in any programming language and any software development process.

The development of a model for an industrial-strength software system, prior to its construction or renovation, is as essential as having a blueprint for a large building. Good models are vital for effective communication among project teams.

In the early 1990s, the tools for OO software modeling emerged, followed by the development of the visual modeling approach. Visual modeling means that you first design your system by drawing diagrams (blueprints) and then employ tools to convert those diagrams into code. The value of such an approach is that the often tedious framework coding is done automatically, freeing the programmer to focus on design issues. The transition from the design to the implementation phase is smoother and more straightforward. Moreover, using the features of reverse engineering and code generation, the developer can move back and forth between the code and the design that is being expressed in the diagrams.

Today, visual modeling tools provide many features that replace some of the more tedious tasks for the designer, programmer, and documentation writer. Some of the leading tools provide so-called round-trip code engineering capabilities – the structure of reverse engineered code is changed in the modeling tool and is generated back without the implementation of specific information (e.g. method bodies, comments) being lost.

MagicDraw is a visual UML modeling and CASE tool with teamwork support. Designed for Business Analysts, Software Analysts, Programmers, QA Engineers, and Documentation Writers, this dynamic and versatile development tool facilitates analysis and design of Object Oriented (OO) systems and databases. It provides the industry's best code engineering mechanism (with full round-trip support for Java, C#, C++, WSDL, XML Schema, and CORBA IDL programming languages), as well as database schema modeling, DDL generation and reverse engineering facilities.

MagicDraw Editions and Features

A detailed list of MagicDraw features can be found at:

<http://www.magicdraw.com/files/brochures/a4/MagicDrawDataSheet.pdf>

MagicDraw Editions

MagicDraw Personal Edition

MagicDraw Personal Edition contains powerful UML diagramming capabilities, including full UML 2 support and extensibility features, basic reporting functionality, and image export. Exported files are stored in XMI format.

All model elements can be accessed via the MagicDraw Open API.

In this edition, you will find everything you need to draw, edit, and publish your UML models.

Personal Edition is available only in a standalone version and is not designed for use with MagicDraw Teamwork Server.

MagicDraw Standard Edition

MagicDraw Standard Edition provides all of the Features of Personal Edition and adds WAE, content, and Robustness diagrams. Standard Edition also adds model analysis and facilitation features, customizable and extendable patterns, integrations with most popular IDEs, and a set of predefined model templates and UML profiles.

Standard Edition supports UNISYS XMI and the latest Model Driven Architecture (MDA) tool offerings. UNISYS XMI diagramming extensions allow the interchange of MagicDraw models with other UML modeling tools.

Standard Edition is available in standalone, floating and mobile license versions and is fully compatible with MagicDraw Teamwork Server.

Standard Edition is ideally suited for analysts and architects who need various model extensions and modeling facilitations.

MagicDraw Architect Edition

The Architect Edition is specially packaged to provide the optimal price and technical features necessary for architects that do not need the full capabilities of the Enterprise Edition. This edition combines the common functionality of the Standard Edition together with some powerful options from the Enterprise Edition. These include: advanced modeling facilitations and analysis, reverse engineering and code generation for DDL, WSDL, CORBA IDL and XML. Architects have less need for IDE integrations as well as Java and C++ code engineering, so these capabilities are not included.

MagicDraw Professional Edition

Professional Edition is built on the Standard Edition capabilities and is available in one of three programming language specific versions-Java, C++ and C#. In addition to the Standard Edition features, Professional Edition adds code generation and reverse engineering functionality. Depending on the language version selected, the user will receive:

- **Java version** - Code engineering for Java, Java bytecode. Integration with Java IDEs.
- **C++ version** - Code engineering for C++.
- **C# version** - Code engineering for C#, CIL (MSIL).

Professional Edition is ideal for anyone who wants to generate code from an existing model or create a UML model from an existing project.

MagicDraw Enterprise Edition

MagicDraw Enterprise Edition represents the top of the line in the MagicDraw family of products, as well as the ultimate solution for all your modeling needs. Enterprise Edition combines all of the functionality of the Personal and Standard Editions, and all three versions of the Professional Edition, into a comprehensive state-of-the-art UML programming solution. But the Enterprise Edition does not stop there, adding code engineering and diagramming functionality in CORBA IDL, EJB, WSDL and XML schema. For working with DB structure, Enterprise Edition not only provides code engineering and diagramming, but also provides structure retrieval via JDBC.

Enterprise Edition is a must when working with multiple development technologies and databases.

The MagicDraw family of award-winning products represents the most powerful and best value in the UML modeling industry today.

MagicDraw Community Edition

MagicDraw Community Edition is free for developers working on non-commercial projects. It has a minimal functionality set and only the class diagram has no limitations. Other diagrams allow saving a project with 25 use cases, 25 states, 25 classifier roles, 25 action states, 25 instances, 25 nodes, and 25 components.

Community Edition is designed for creating static structure models when XML output is needed and it is ideally suited for Open Source projects.

Printing and image export capabilities are also included.

MagicDraw Reader Edition

MagicDraw Reader Edition is made for reading and previewing UML models created with MagicDraw and it is free of charge. It is extremely useful when you want to share your ideas expressed in UML with partners, colleagues, or clients, who do not have a copy of MagicDraw. Printing and image export capabilities are also included.

Since MagicDraw version 14.0, Reader Edition has the ability to open and review Teamwork Server projects.

MagicDraw Features

Reports Generation

You will find a complete description of the MagicDraw Report Wizard, related OpenAPI, and tutorial in *MagicDraw ReportWizard UserGuide.pdf*.

Floating License Server

The Floating license agreement does not limit the number of clients you can install on different computers. It only limits the number of applications that can run at the same time. To control loaded applications, a server is required. The server can be installed on several computers, but simultaneously can be started only on the one of them. The license key of the floating server provides information to the server about how many applications may run simultaneously for the particular MagicDraw edition. If you upgrade the MagicDraw version, you do not have to obtain a new license key for the server. The Administrator's Console is used to manage client connections and configure the server.

For more information about MagicDraw Floating License Server, see “MagicDraw FloatingLicense UserGuide.pdf”.

Teamwork License Server

NOTE The Teamwork License Server is available with MagicDraw application (Standard, Professional, Architect, and Enterprise editions).

With MagicDraw Teamwork Server, you can assign as many developers as needed to work simultaneously on the same project using multiple workstations. The resulting Teamwork project is saved on the server for sharing by other MagicDrawTM applications. Users with administrator rights can create new users by giving them their own login name and various permissions to work on projects. Depending on permissions, users can update, commit, edit, create, and delete model elements, diagrams, and projects.

To enable Teamwork support, you should install and run the MagicDraw Teamwork Server software. Each MagicDraw application acts as a client in the Teamwork system.

Teamwork Server functionality is available with MagicDraw client Standard, Professional, and Enterprise Editions only. MagicDraw Reader Edition is allowed to open and review teamwork projects.

For more information about Teamwork License Server, see “MagicDraw Teamwork UserGuide.pdf”.

Code and Database Engineering

MagicDraw code engineering provides a simple and intuitive graphical interface for merging code and UML models, as well as preparing both code skeletons out of UML models and models from code.

MagicDraw code engineering features can be very useful in several situations:

- You already have code that needs to be reversed to a model.
- You wish to have the implementation of the created model.
- You need to merge your models and code.

The tool may generate code from models and create models out of code (reverse). Changes in the existing code can be reflected in the model, and model changes may also be seen in your code. Independent changes to a model and code can be merged without destroying data in the code or model.

MagicDraw UML code engineering supports Java, Java Bytecode, C++ (ANSI, CLI, Managed), C#, CIL, CIL Disassembler, CORBA IDL, DDL (Cloudscape, DB2, Microsoft Access, Microsoft SQL server, MySQL, Oracle, Pervasive, Pointbase, PostgreSQL, Standard SQL, Sybase), XML Schema, WSDL, and EJB 2.0 notation.

For more information on working with code engineering and databases, see “MagicDraw Code&DatabaseEngineering UserGuide.pdf”.

OpenAPI

This document describes the MagicDraw Open Java API and provides instructions on how to write your own plug-ins, create actions in the menus and toolbars, change UML model elements, and create new patterns.

For more information on working with OpenAPI, see “MagicDraw OpenAPI UserGuide.pdf”.

Integrations

MagicDraw supports the following integrations:

- Eclipse
- RAD
- BEA Workshop
- IntelliJ

- JBuilder
- NetBeans
- OAW
- AndroMDA
- ProActivity
- CVS
- CaliberRM

For more information about MagicDraw integrations, see “MagicDraw Integrations UserGuide.pdf”.

MagicDraw Customization

MagicDraw introduces several advanced customization engines, based on UML Profiles:

- Custom Diagram Wizard allows creating your own diagram types for custom profile. You may include your own toolbars, stereotyped elements, symbol styles, and custom smart manipulators. Such customization is saved in the special “diagram descriptor” that could be exchanged between users. This allows others to use your custom diagrams.
- Domain Specific Language Customization Engine (DSL customization engine) allows “tuning” domain specific profiles, customizing multiple GUI, model initialization, and semantic rules, creating your own specification dialogs. DSL customization is model-driven approach, based on UML profiling. Customization is saved as a UML model.
- Advanced UML Profiling allows the use of some profiling enhancements that are not defined in UML, but helps to solve some common problems like tag grouping, unwanted stereotypes, tags hiding, etc.

For more information about MagicDraw customization see “UML Profiling and DSL UserGuide.pdf”.

MagicDraw Plugins

For the full MagicDraw plugins list, see the following website: www.magicdraw.com/plugin.

DoDAF

This document describes all DoDAF artifacts, their forms (diagram, report, matrix, and table), mapping to MagicDraw UML model, implementation details, and useful tips. This information is provided for every type of DoDAF product.

For information on how to install, set up, and use the DoDAF plugin, see “DoDAF Plugin User-Guide.pdf”.

SysML

SysML (Systems Modeling Language) supports the specification, analysis, design, verification and validation of a broad range of complex systems.

The SysML plugin for MagicDraw supports all SysML diagrams, including Requirements, Block Definition, Internal Blocks, Parametric, and others. With SysML, MagicDraw adds support for additional specification, analysis, design, and validation of a broad range of systems and system integrations.

For information on how to install, set up, and use the SysML plugin, see “SysML Plugin User-Guide.pdf”.

MagicRQ

MagicRQ plugin is the MagicDraw integration with IBM/Rational® RequisitePro® and Telelogic DOORS requirements and use case management tools.

For information on how to install, set up, and use the MagicRQ plugin, see “MagicRQ Plugin User-Guide.pdf”.

NOTE

For information about MagicDraw and Plugins compatibility see the following website: <http://www.magicdraw.com/compatibility>

The table shows which versions of MagicDraw and MagicDraw Plugins can work together.

MagicDraw RConverter

MagicDraw RConverter generates data files in Rational Rose's (*.mdl) to MagicDraw's (*.xml). By obtaining information from Rational Rose via Rational Rose API (REI) and using VB6.0 to calculate the change, the resulting file is saved in the MagicDraw file (xml file extension) format.

For information on working with RConverter, see “NM_MagicDraw_RConverter_UserManual.pdf”.

MagicDraw Welcome Screen

The Welcome screen is displayed in the MagicDraw desktop when no projects are opened. It helps to manage projects, provides quick access to the product descriptions, samples, the latest news and updates. See the Welcome screen in Figure 1 on page 30.

To open the Welcome screen from the **Help** main menu, choose the **Show Welcome Screen** command (note that no projects should be opened).

The **Show Welcome Screen** option is added to the **Environment Options** dialog box, **General** branch, **Display** group. Using this option you can set whether the Welcome Screen will be displayed.

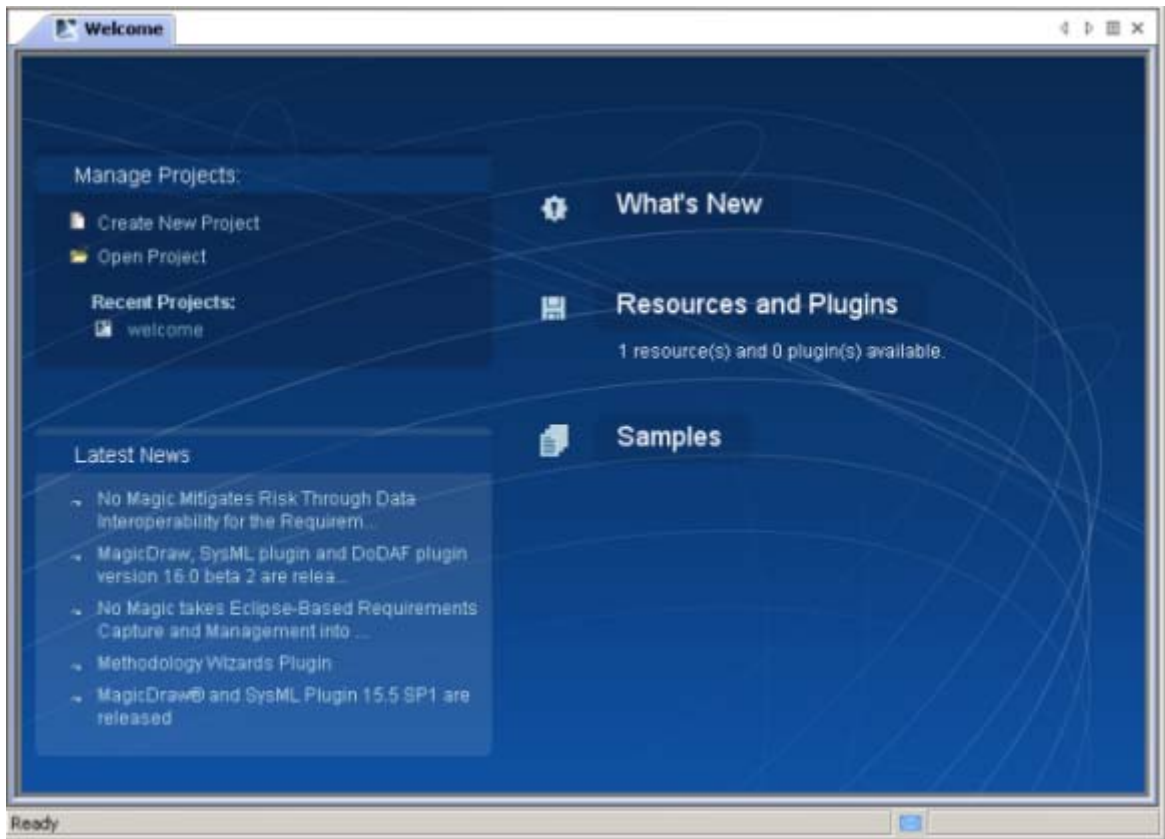


Figure 1 -- The MagicDraw Welcome Screen

Manage Projects

Click **Create New Project**, to create a blank project.

Click **Open Project**, to open existing project.

The **Recent Projects** list contains list of the recently opened projects.

Latest News

Click on the particular news to read more detailed description. The news will be displayed in the MagicDraw News Reader. For more information, see “MagicDraw News Reader” on page 35.

What's New tab

Click the description to open the following items:

- Introduction to MagicDraw. The introductory MagicDraw sample is opened.
- New and noteworthy. The www.magicdraw.com/newandnoteworthy webpage is opened. See the list of a new MagicDraw features for the newest version.
- MagicDraw news. The MagicDraw News Reader is opened. For more information, see “MagicDraw News Reader” on page 35.
- MagicDraw updates. The **Updates Information** dialog box opens, with a list of the latest updates. Here you will be able to see what updates are available and update to a newer version.
- MagicDraw home page. Click this link to navigate to the www.magicdraw.com webpage.

1 INTRODUCING MAGICDRAW

MagicDraw Welcome Screen

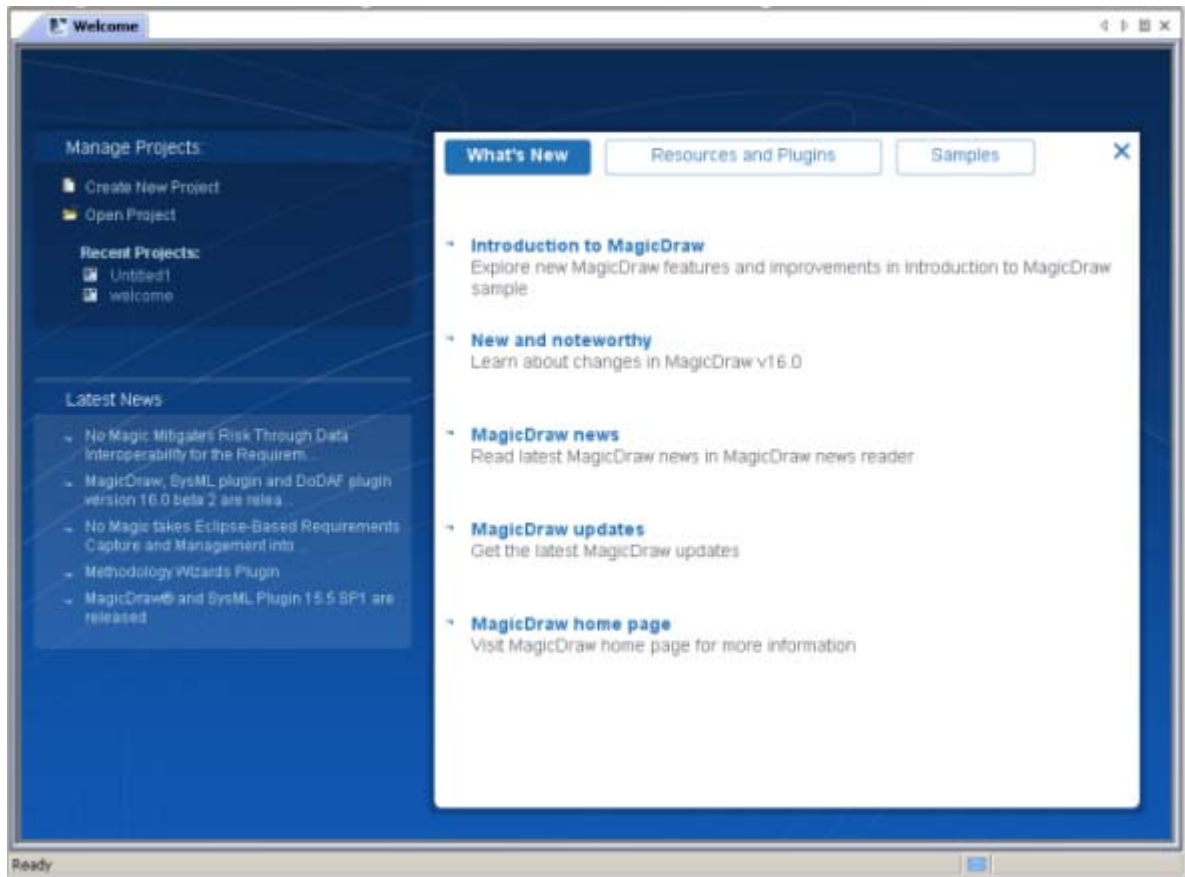


Figure 2 -- The Welcome Screen, What's New tab

Resources and Plugins

- Review available plugins. The www.magicdraw.com/plugin page is opened. All available plugins descriptions are available in this page.
- Install available resources. Click to open the **Resource/Plugin Manager** window. Download and install available resources and plugins using this manager. For more information, see "Resource Manager" on page 497.
- MagicDraw manuals. Click to open the <MagicDraw installation directory>\manuals folder, which contains MagicDraw manuals.

1 INTRODUCING MAGICDRAW

MagicDraw Welcome Screen

- Online demos. The www.magicdraw.com/viewlets webpage is opened. View demos, which introduces MagicDraw and MagicDraw features.
- MagicDraw eSchool. The <http://school.nomagic.com/> webpage is opened.

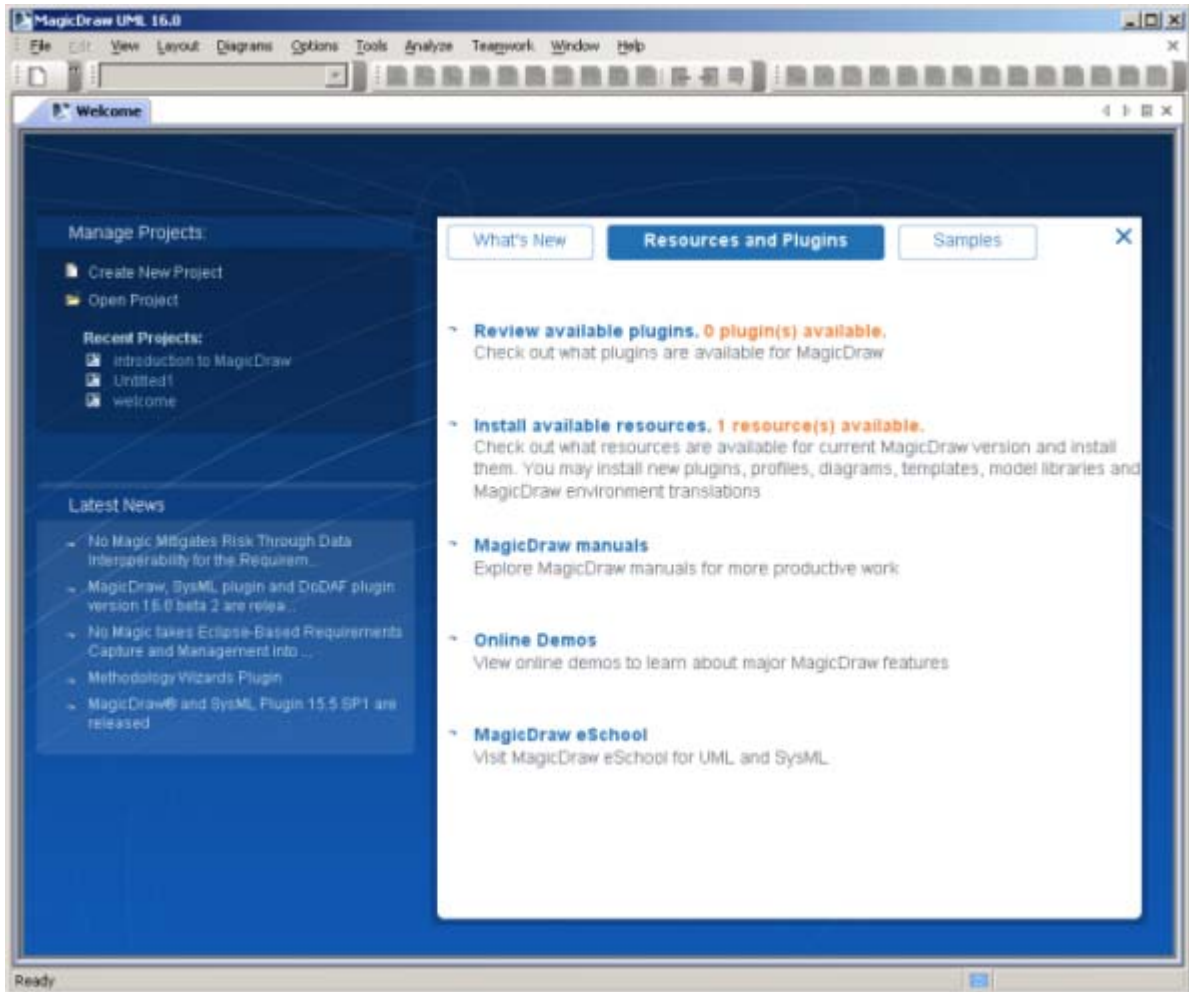


Figure 3 -- The Welcome Screen, Resources and Plugins tab

1 INTRODUCING MAGICDRAW

MagicDraw Welcome Screen

Samples

Quick navigation to MagicDraw samples. All samples from <MagicDraw installation folder> \samples directory are accessible in this page.

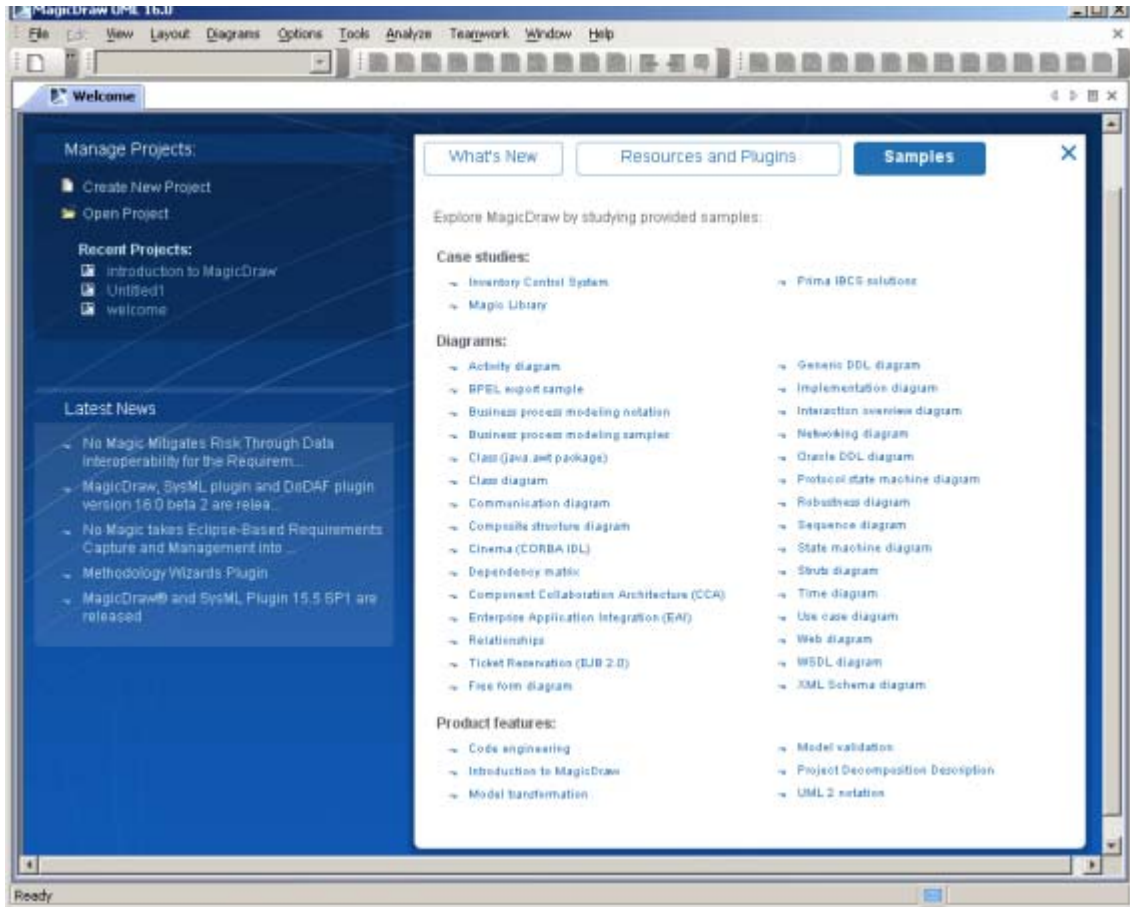


Figure 4 -- The Welcome Screen, Samples tab

MagicDraw News Reader

Information about the latest MagicDraw events is provided in the new MagicDraw News Reader. The News Reader is accessible from the MagicDraw **Help** main menu, the **MagicDraw News Reader** command (see Figure 5 on page 36).

MagicDraw News Reader informs about:


- No Magic News. All news regarding company news, product news, new services provided, etc.
- New Versions. Messages about new MagicDraw releases, betas, service packs, plug-ins.
- Resources. Messages about MagicDraw resources updates.


When some news is available, a small envelope icon will be displayed at the right of MagicDraw status bar. Click on this icon to invoke the MagicDraw News Reader (see Figure 6 on page 37).

Reading news in the MagicDraw News Reader

Select the news channel at the left side of the MagicDraw News Reader and then select one of the news in the list. Below the title of the selected news, the description is presented. Unread messages are displayed in bold.

Click the **Open in Browser** link to read description on www.magicdraw.com website.

To refresh news, click the  **Refresh** button at the left top corner of the **MagicDraw News Reader** window.

To mark all the selected items as read, click the  **Mark item as read** button.

Setting options of the MagicDraw News Reader

To change the checking period, change the **Check for MagicDraw News** property in the **Environment Options** dialog box, **General** section. Property provides the following options:

- Once a day
- Once a week

1 INTRODUCING MAGICDRAW

MagicDraw News Reader

- Once a month
- Do not check.

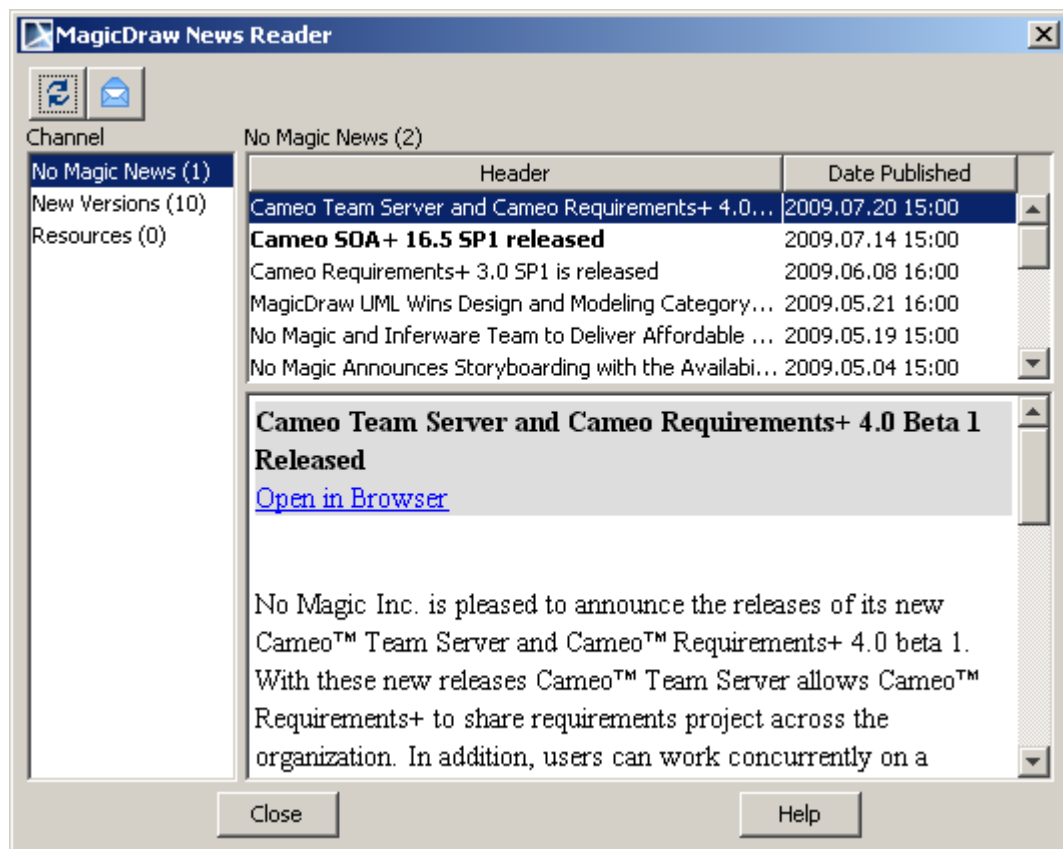


Figure 5 -- MagicDraw News Reader

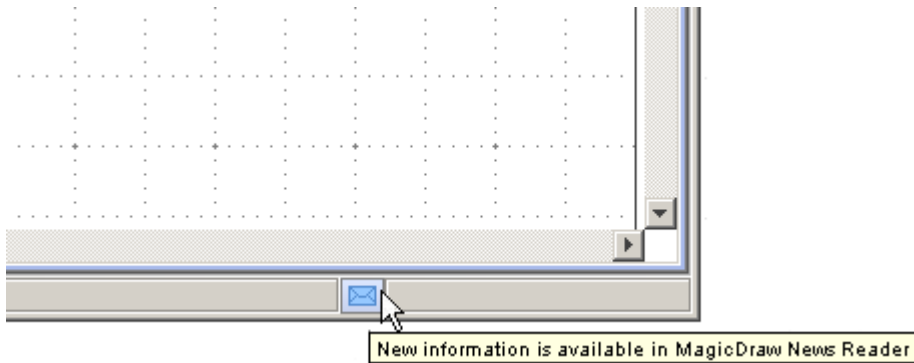


Figure 6 -- Envelope icon informs about news in MagicDraw News Reader

MagicDraw Documentation and Support

MagicDraw provides several kinds of documentation. Choose the way you want to learn.

The main source of information about MagicDraw is www.magicdraw.com and documentation can be downloaded from www.magicdraw.com/documentation.

New and Noteworthy

For information on MagicDraw new features, see New and Noteworthy at <http://www.magicdraw.com/newandnoteworthy>.

Manuals and User Guides

You can find the MagicDraw manual and user guides in "<MagicDraw installation>\manual" folder.

Help

The integrated help within MagicDraw is based on JavaHelp. MagicDraw help provides detailed descriptions of all MagicDraw dialog boxes, commands, and shortcut menus. You will also find a How-to list, as well as main descriptions and examples of all UML model elements.



Hints Associated to User's Actions

MagicDraw now provides hints to help you MagicDraw (Figure 7 on page 38). Hints related to your actions will open and inform you about the functionality that is available in MagicDraw and show you how to perform some operations more effectively.

Hints are displayed in the lower right-hand corner of the MagicDraw application. Figure below shows an example of a hint.

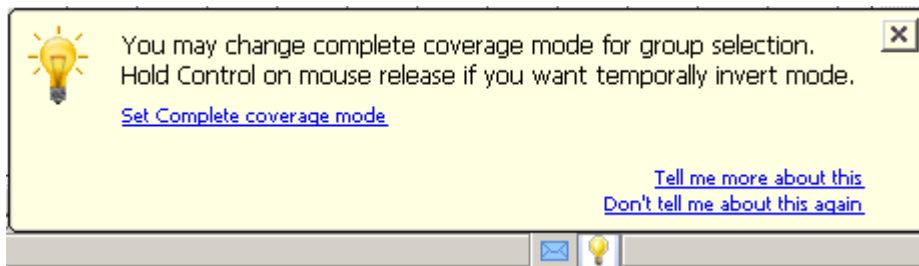



Figure 7 -- Hints Associated to User's Actions

Hint is displayed for a 10 seconds or for a period while mouse is rolled on hint. Small icon  will be displayed in a status bar after hint will hide. Clicking on this icon, you can open hint again.

Changing hints display mode

To change hints display mode:

- Change the **Hints display mode** property in the **Environment Options** dialog box, **General** branch, **Display** group.

The **Hints display mode** property specifies whether hints on MagicDraw functionality related to user actions will be displayed. Custom hint set does not include hints, that are asked not to be shown by user. Select option **Display all hints again** to reset custom hint set to show all hints. Selecting value **Show all hints again** will delete the list of the hints that should not be displayed.



Productivity Tips Displayed in Progress Window

MagicDraw now displays productivity tips in a progress window whenever it performs a long task (Figure 8 on page 39).

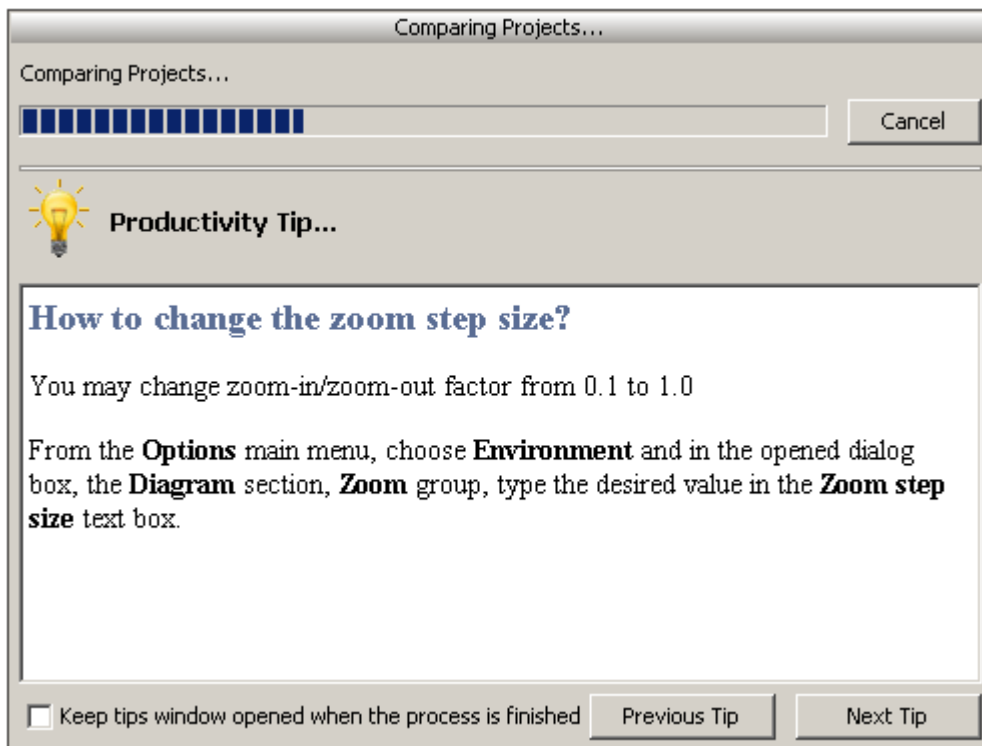


Figure 8 -- Productivity Tip

To show or hide productivity while running long task:

Select or clear the **Show tips while running long tasks** property in the **Environment Options** dialog box, **General** branch, **General** group.



Tutorials

New interactive diagrams tutorials for new MagicDraw and UML users are now available. The tutorials provide step-by-step instructions on how to work with UML diagrams and start modeling with MagicDraw. The tutorials also provide you with lots of links to other MagicDraw documentation and worldwide tutorials sources. The tutorials include:

new diagram creation, Class diagram, Use Case diagram, Activity diagram, Sequence diagram and other diagram tutorials.

To open the Quick Start tutorials:

1. From the **File** menu, select **New Project**. The **New Project** dialog will open (Figure 9 on page 41).
2. Select the **Project from Template** icon on the left-hand side of the **New Project** dialog.
3. Select **Guide to UML Diagrams Project > Guide to UML Diagrams Project** in the Select template tree.
4. Click **OK**. The MagicDraw project with its tutorials will open (Figure 10 on page 42).

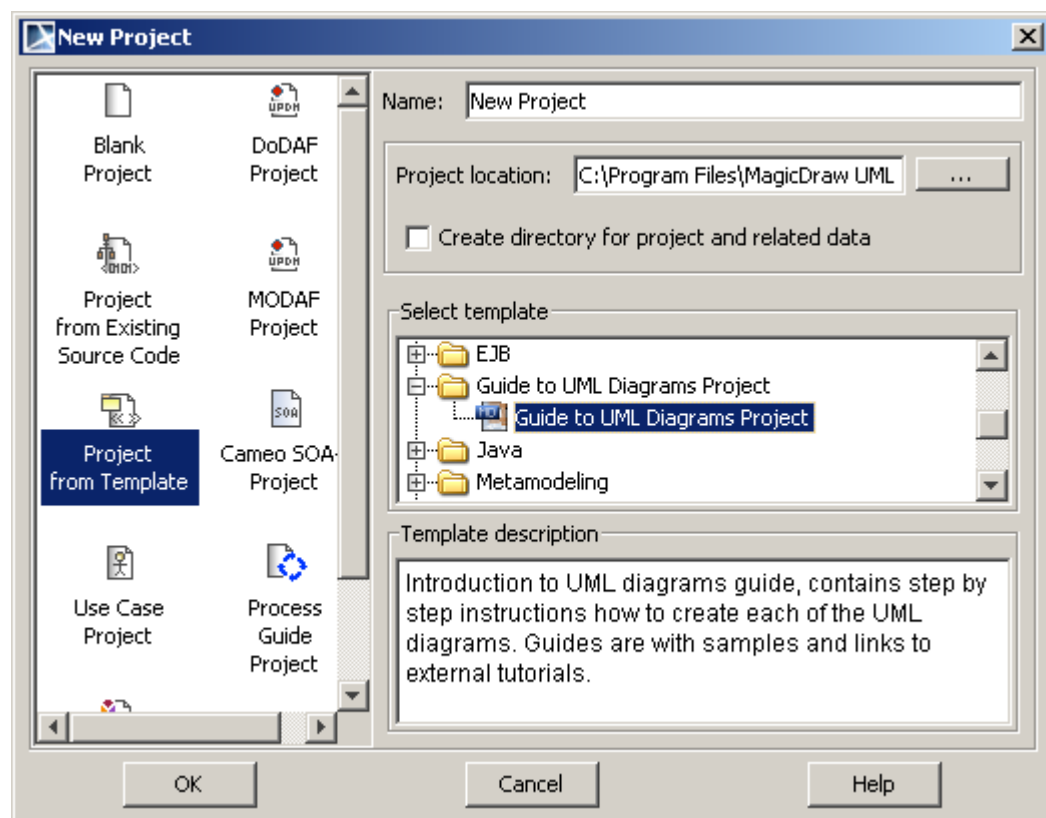


Figure 9 -- The New Project Dialog

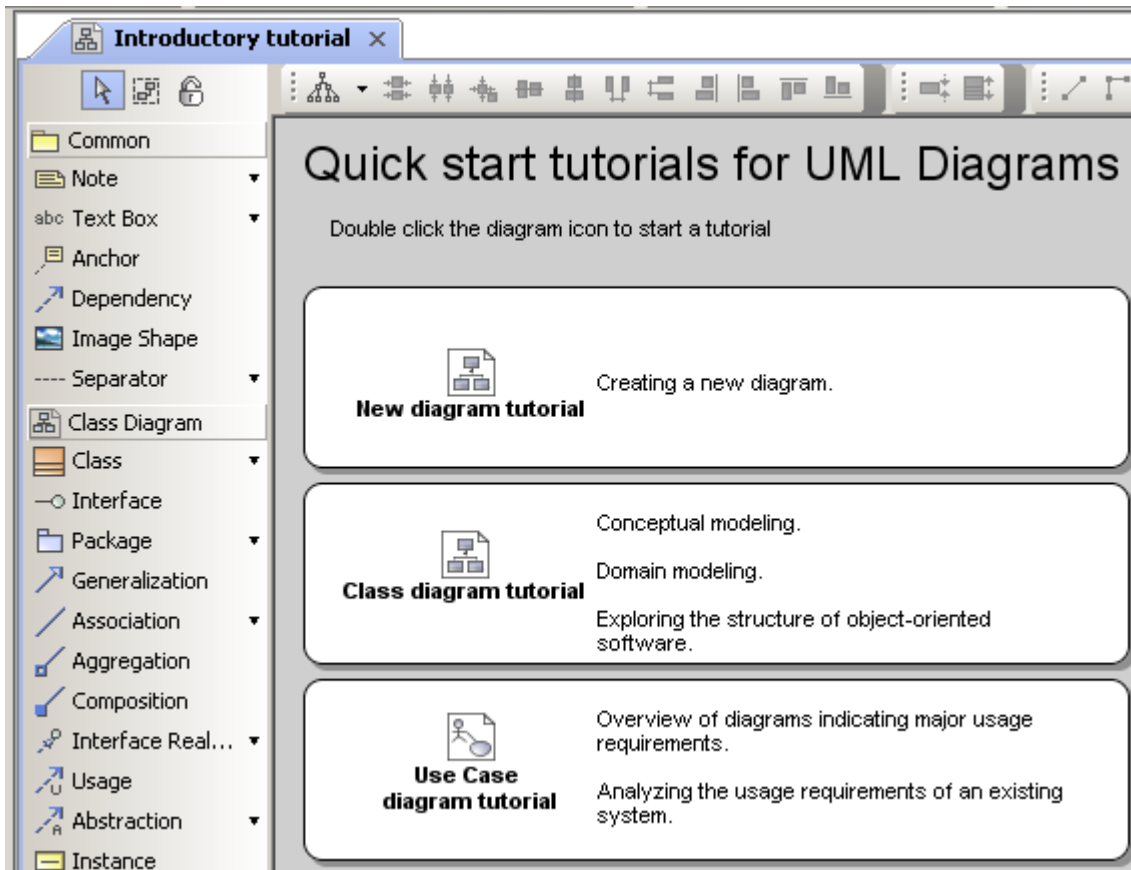


Figure 10 -- Quick Start Tutorials for UML Diagrams

Other Documentation

- The main [readme.html](#) and [notes.html](#) files are located in the main MagicDraw installation directory. Readme documents are also available for MagicDraw integrations. In the notes.html file you will find the main information about MagicDraw features.
- *Viewlets*. View online demos introducing MagicDraw and particular functionalities. You may find online demos at www.magicdraw.com/viewlets.
- *Samples*. In the MagicDraw installation directory (samples directory), you will find the samples of MagicDraw projects.

Support

FAQ

Before you call or write an email, look into our [FAQ](#) (Frequently Asked Questions) list. It is constantly updated and perhaps it contains an answer to your question.

Newsgroups

Discuss and get answers about MagicDraw in [newsgroups](#).

E-mail

Professional e-mail support is provided for registered MagicDraw users.

support@magicdraw.com - private questions about product installation, and more.

sales@magicdraw.com - questions regarding academic and site discounts and/or quotes.

contact@magicdraw.com - e-mail for other contacts.

Get an answer in 24 hours!

Bug Reports

Submit notifications of software errors by submitting a bug directly from MagicDraw (Help menu->Submit a Bug) or by sending an email to support@magicdraw.com.

These reports help us address issues in a more timely manner, as well as speeding up maintenance releases that are free of known defects.

When sending your report by email, please include (if applicable):

- MagicDraw UML version number and name of the edition (Reader, Community, Personal, Standard, Enterprise, Professional Java, Professional C++, Professional C#, Demo, or Academic).
- demo CD, our homepage download.
- Version distribution (Standard/Professional, Windows/Java bytecode).

- Your OS name and version.
- JDK version, JVM vendor.
- CLASSPATH settings.

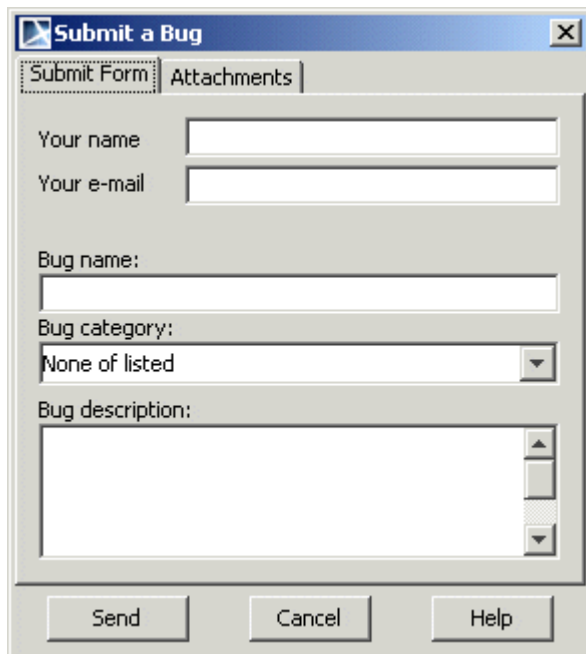
If you do not have this information, you can find it by opening the **About** dialog from the MagicDraw **Help** menu. If MagicDraw is unable to load your file, and the file is not confidential, please attach it to your submission.

This will assist us in our analysis of your problem.

To open the **Submit a Bug** dialog box

Select **Submit a Bug** from the **Help** menu.

NOTE If you are already registered personal information will be filled in to bug submit dialog.



The image shows a screenshot of the 'Submit a Bug' dialog box. The dialog has a title bar with the text 'Submit a Bug' and a close button (X). Below the title bar, there are two tabs: 'Submit Form' (which is selected) and 'Attachments'. The 'Submit Form' tab contains several input fields: 'Your name' (a text box), 'Your e-mail' (a text box), 'Bug name:' (a text box), 'Bug category:' (a dropdown menu with 'None of listed' selected), and 'Bug description:' (a large text area with vertical scrollbars). At the bottom of the dialog, there are three buttons: 'Send', 'Cancel', and 'Help'.

Figure 11 -- Submit a Bug dialog box, Submit Form tab

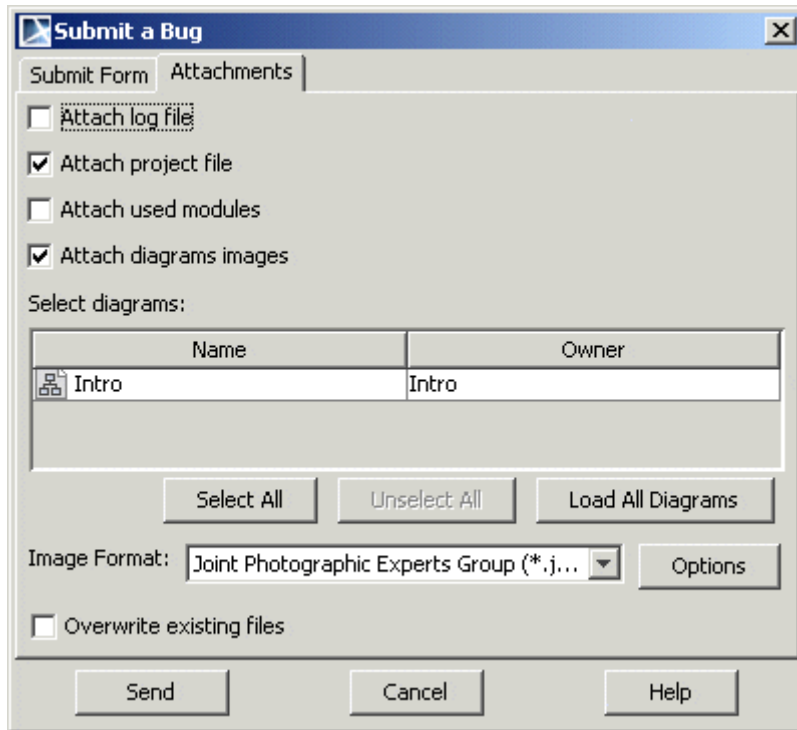


Figure 12 -- Submit a Bug dialog box, Attachments tab

Tab name	Box	Function
Submit Form Fill in the information about yourself and describe the bug or suggestion.	Your Name	Type your name.
	Your E-Mail	Type your e-mail address.
	Bug Name	Give a name to the bug.
	Bug Category	Select a bug category, or create a new category that is more fitting.
	Bug Description	Describe the bug or suggestion.

Tab name	Box	Function
Attachments Select a file or diagram you want to attach to the bug description.	Attach log file	Attach MagicDraw log file with included errors.
	Attach project file	Attach current MagicDraw project and send it to the support team.
	Attach used modules	Attached modules used in the project.
	Attach diagram images	Attach images of selected diagrams.
	Image format	Select format of the diagram images, which will be sent to the support team. To specify image options click the Options button.
	Overwrite existing files	Select the Overwrite existing files check box, to overwrite existing files with the current one.
	Select All	Select all diagrams displayed in the Select Diagrams list.
	Clear All	Remove all previously selected diagrams from bug report.
	Load All Diagrams	Loads all diagrams of the project. By default only open diagrams are shown in the Select Diagrams list.
Buttons	Send	Send the bug. RECOMMENDATION: You may submit a bug by mailing to support@magicdraw.com, but it is recommended that you submit a bug directly from within the MagicDraw tool. This assures that the MagicDraw team will receive the description of the bug along with your system information.
	Cancel	Exits the dialog box without saving changes.
	Help	Displays MagicDraw Help.

To submit a bug when application is unresponsive

If MagicDraw becomes unresponsive, a separately executable tool is provided for analyzing the status of the process to aide in bug submission. In these situations, manually start the submitbug.exe file (located in the <MagicDraw installation directory>\bin folder) and follow directions. After submit-

bug.exe is started, the **Submit a Bug** dialog box opens. For more information about the **Submit a Bug** dialog box, see “To open the Submit a Bug dialog box” on page 44.

View and submit internal errors

If an error occurs, an error message will appear at the bottom of the MagicDraw window.

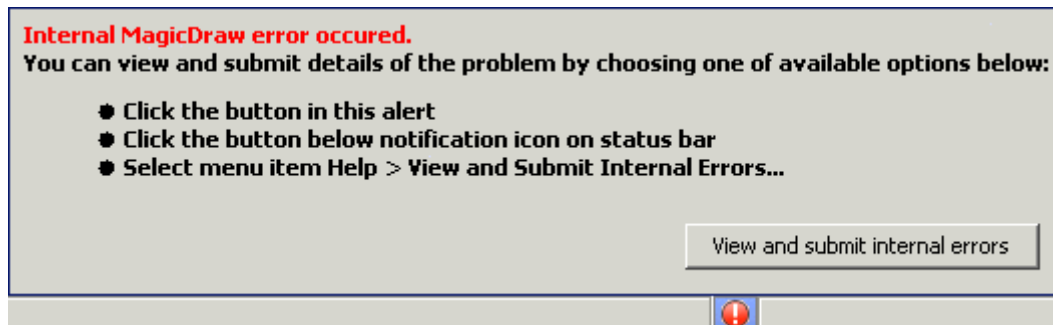



Figure 13 -- Message, about MagicDraw internal error.

To view internal errors

To view internal errors you have to open the **Unhandled Errors** dialog box. There are three methods for to open the **Unhandled Errors** dialog.

- Click the **View and submit internal errors button** in the message window.
- From the **Help** main menu, choose the **View and Submit internal errors** command.
- Click the notification icon  on the status bar.

Note: The **View and Submit internal errors** command is only active in the **Help** menu and the red button at the bottom of the status bar only exists if the **Submit errors** dialog box contains errors.

To submit an error

1. Open the **Unhandled Errors** dialog box.
2. Click the **Submit** button in the **Unhandled Errors** dialog box. The **Submit Error** dialog box appears.
3. Fill in **Your name**, **Your e-mail** and **Bug description** fields and click the **Send** button. The error will be sent to the MagicDraw support team.

2 GETTING STARTED

In Getting Started, you will find information about how to install, update, and configure MagicDraw.

1. "System requirements", on page 48
2. "Java Virtual Machine (JVM)", on page 49
3. "Installation Procedure", on page 50
4. "MagicDraw Configuration Files Location", on page 53
5. "User Registration", on page 54

System requirements

To run MagicDraw UML, your system must meet the following requirements:

Resource type	Minimum	Recommended
CPU	PentiumTM 3, 800 MHz	PentiumTM 4, 1.4 GHz or higher
Memory	768 MB	1 GB RAM recommended; more memory generally improves responsiveness. 2 GB RAM is recommended for MicrosoftTM Windows Vista and for very large projects
Disk space	400 MB	400 MB or more
Video mode	800*600 @ 64k Colors	1280*1024 @ 64k Colors
Operating system	All that have Java compatible JVM 5 or 6: Windows 95/98/NT/2000/XP/Vista, Linux, Mac OS X (most testing procedures and debugging were performed on these platforms)	
HTML browser	Any	Netscape Navigator or IE
Java Virtual Machine (JVM)	JDK 5	Sun's JDK 1.6.0_15

Java Virtual Machine (JVM)

You may have the JVM installed on your computer or install JVM together with MagicDraw specifically for the MagicDraw application. JVM is an application that provides the interpretation of the Java bytecode Java class files. Different operating systems may have different JVM implementations, therefore some bugs may be specific to the operating system or JVM.

MagicDraw is a stable environment, if it is configured properly and if a stable JVM is used. **USE THE RECOMMENDED JVM TO AVOID MOST PROBLEMS.** All recommendations are written below. Because MagicDraw is a Java application, most of the stability and performance depends on the JVM implementation. Refer to the JVM specification and problems description, if you have problems on a specific OS.

JDK 1.1.x, 1.2.x, 1.3.x, and 1.4.x are not supported.

You can review your system and the JVM information in the **About** dialog box, **Environment** tab (from the **Help** menu, select **About**). The JDK version can also be encountered from the command line by typing:

```
java -version
```

To change Java version

All platforms:

Change line in file `mduml.properties` (this file is in `<MagicDraw installation directory>/bin` folder):

`JAVA_HOME = <path to new JDK>`

NOTE

Integrated MagicDraw runs on the JVM specified by the IDE. In order to change the JVM, you need to modify the startup properties for the IDE that MagicDraw integrates with. If you are running MagicDraw integrated with IDE, read the appropriate `readme.html` for specific integration. This file can be found in `<MAGICDRAW installation directory>/integrations/<IDE directory>`.

To redirect output to the console instead of a log file

By default all MagicDraw output info goes into a file. The output can be redirected to the console instead of a log file.

All platforms:

Pass application argument (not java property) `-verbose`.

Add argument into the JAVA_ARGS line in the mduml.properties file (this file is in <MagicDraw installation directory>/bin folder):

APP_ARGS =-verbose

Operating System - dependent issues

Because MagicDraw is 100% Pure Java, it is platform independent and runs on a variety of operating systems. However, for Java applications to interact properly with the operating system, Java Virtual Machine (JVM) software is required. JVM software varies depending on the operating system and MagicDraw may perform inadequately with the wrong one.

The performance of Java applications depends on the performance of the Java Virtual Machine. A badly designed Machine may degrade the performance of MagicDraw. It could even cause MagicDraw to fail or crash. To avoid these problems, we recommend that you use the following Virtual Machines:

- Sun (JDK standard) for Solaris and Linux. JDK 1.6.0_15 is recommended.
- Sun (JDK standard) for Windows (95/98, 2000, 2003, NT, Vista). JDK 1.6.0_15 is recommended.
- We recommend using Mac OS X Leopard or Mac OS X Snow Leopard, and Java 1.5.0_19 for 32-bit Mac OS X, and Java 1.6.0_10 for 62-bit Mac OS X.

Installation Procedure

First, obtain the MagicDraw installation files. You can download the latest version from the MagicDraw homepage (<http://www.magicdraw.com/>.) Because MagicDraw is a Java application, you will need more than the installation files to run the tool successfully. You can also install JVM together with MagicDraw or you may already have it installed. Information about the latest Java ports is available at http://www.magicdraw.com/jvm_list.htm.

NOTE If an installation is for Windows, and has a JVM you do not need anything else.

Windows 2000/2003/NT/XP/Vista*

After downloading, double-click MD_UML_<version>_win.exe.

The Setup Wizard automatically adds MagicDraw UML shortcuts to the start menu and the desktop. You may also execute the shortcuts from the installation directory.

NOTE MagicDraw is working on Vista OS since MagicDraw 12.5 version.

Unix

After downloading, open a shell and go to the directory where you downloaded the installer. At the prompt type: `sh ./MD_UML_<version>_unix.sh`.

MAC OS X

Recommended Mac OS X Leopard, Mac OS X Snow Leopard, and Java 1.5.0_19 for 32-bit Mac OS X, and Java 1.6.0_10 for 62-bit Mac OS X.

After downloading, double-click `MD_UML_<version>_mac.dmg` and install the application by dropping the launcher to the Applications folder (or somewhere else).

We suggest to create additional folders for different MagicDraw versions in MAC OS X Applications folder in case do not overwrite old client with the new one. After installing MagicDraw to several folders you will be able to import the old MagicDraw version configuration.

To install MagicDraw to newly created folder:

1. In the Applications folder create folder named as MagicDraw version. For example, MagicDraw UML 16.0.
2. Drag `MD_UML_<version number>_mac.dmg` content to the Applications/MagicDraw UML 16.0.

All other platforms instructions (no install version)

Download `MD_UML_<version>_no_install.zip`.

After downloading, extract the contents of the zip. These files require some modifications prior to launching them.

On Windows platform:

- Launch `mduml.exe` in bin directory to start MagicDraw.

On Unix or Mac OS X platform:

- Launch `./mduml` in bin directory to start MagicDraw.

Be sure you have JVM installed.

Licensing Information Display

Information about installed or needed licenses and the status is presented in the MagicDraw **About** screen with the ability to remove unused licenses (see Figure 14 on page 53).

User ID

User ID is displayed in the **About** dialog box, **Licensing** tab. Please refer to user ID when contacting support or sales.

Not installed indication

If license is not installed, "Not installed" text is displayed after the license name.

Not started indication

If licenses is installed but not started "Not started" text is displayed after license name and reason is be given. Possible reasons:

- Required resource is not installed.
- Plugin startup failed.
- Other.

The Remove Unused Licenses button

Press button **Remove Unused Licenses** in the **About** screen, **Licensing** tab to remove licenses of not installed plugins.

NOTE

2 GETTING STARTED

MagicDraw Configuration Files Location

If you have any questions or issues, please submit a bug using the Submit a Bug dialog box. For more information, see “View and submit internal errors” on page 47.

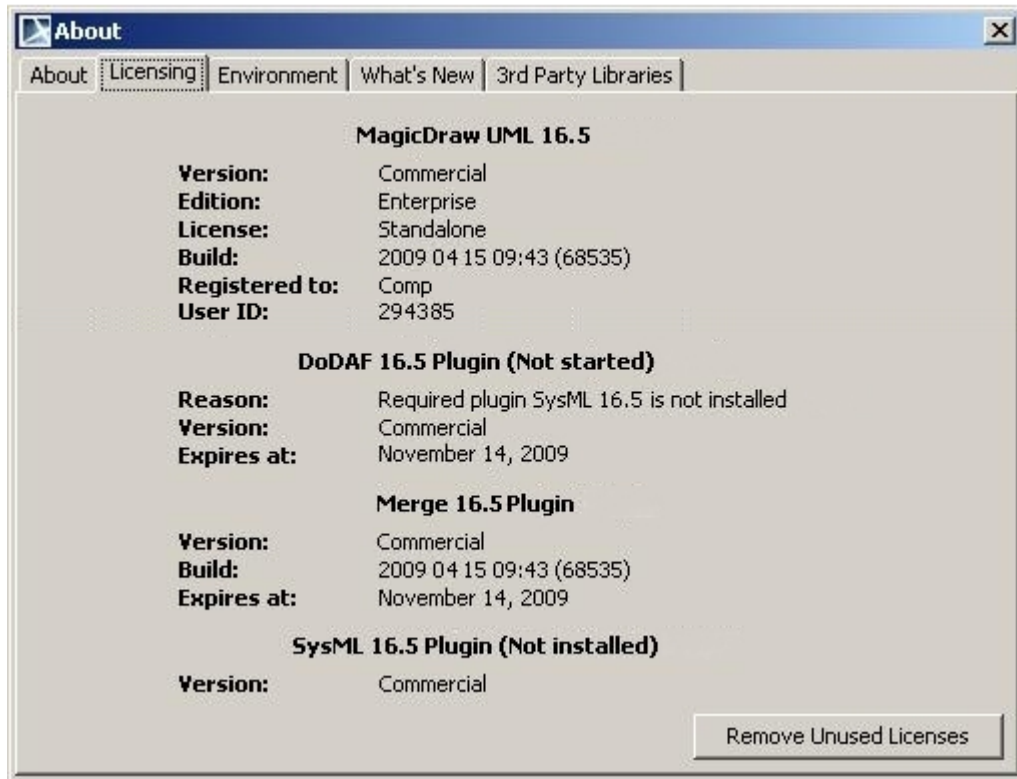


Figure 14 -- The MagicDraw About window, Licensing tab

MagicDraw Configuration Files Location

By default MagicDraw configuration and auxiliary files are stored in the user home directory - <User home directory>/..magicdraw/<version>.

You may also save configuration files:

- In the MagicDraw installation directory.
- In your chosen directory.

You may check the exact path to the configuration files in the MagicDraw **About** screen, **Environment** tab, **Configuration files** line (invoke the About screen from the **Help** main menu, **About MagicDraw** command).

To store MagicDraw configuration files in the MagicDraw installation directory

Add argument to JAVA_ARGS line in the file mduml.properties file (this file is in <MagicDraw installation directory>/bin):

```
JAVA_ARGS=-DLOCALCONFIG=false
```

To store MagicDraw configuration files to your chosen location

You may define custom path for configuration files in two ways:

- In the mduml.properties file, JAVA_ARGS line add the property
`-Dlocalconfig.location=<custom path>`
- Define custom path in a newly created file:
 1. In the <User home directory>/magicdraw/<version> folder, create a file named *magicdrawredirect*.
 2. In the created file, type the absolute path where the MagicDraw configuration and auxiliary files will be saved, for example: *C:\<directory name>*.

NOTE

If MagicDraw is configured to store files in the MagicDraw installation directory (see the topic “To store MagicDraw configuration files in the MagicDraw installation directory” above), files will not be stored to your chosen location.

User Registration

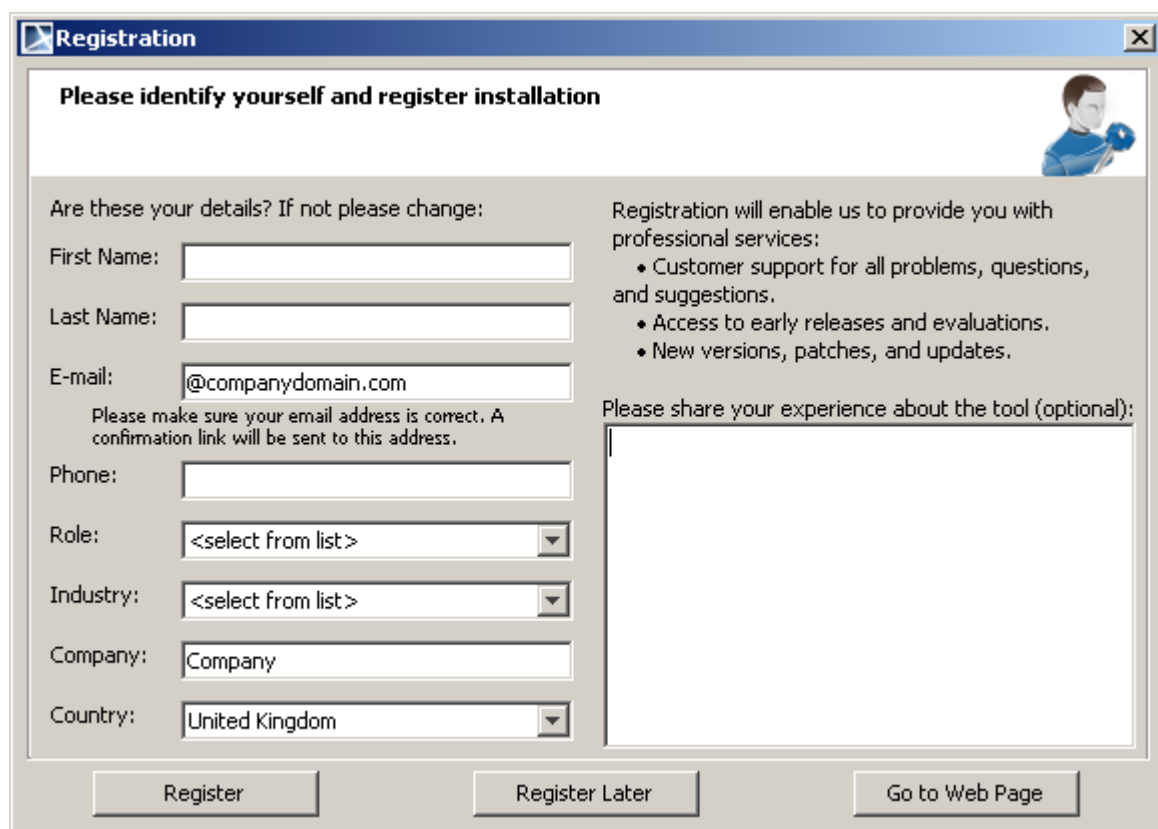
User Registration allows you to access dedicated resources on MagicDraw Website such as dedicated online support, answers database, new products evaluation, and beta products.

2 GETTING STARTED

User Registration

The **Registration** window will open the first time MagicDraw is started (Figure 15 on page 55). You can complete Product Registration at any time by selecting **Register...** on the MagicDraw **Help** menu.

NOTE No Magic, Inc. respects your privacy. We will only use your personal information for communications and management of your online account, and the products you register with your account.



The Registration dialog box has a title bar with the word "Registration" and a close button. The main area is titled "Please identify yourself and register installation" and features a small icon of a person with a magnifying glass. The form is divided into two columns. The left column contains a heading "Are these your details? If not please change:" followed by input fields for First Name, Last Name, E-mail (with a placeholder "@companydomain.com"), Phone, Role (a dropdown menu showing "<select from list>"), Industry (a dropdown menu showing "<select from list>"), Company (with the text "Company"), and Country (a dropdown menu showing "United Kingdom"). Below these fields is a note: "Please make sure your email address is correct. A confirmation link will be sent to this address." The right column contains a heading "Registration will enable us to provide you with professional services:" followed by a bulleted list: "• Customer support for all problems, questions, and suggestions.", "• Access to early releases and evaluations.", and "• New versions, patches, and updates." Below the list is a heading "Please share your experience about the tool (optional):" and a large text area. At the bottom of the dialog are three buttons: "Register", "Register Later", and "Go to Web Page".

Registration

Please identify yourself and register installation

Are these your details? If not please change:

First Name:

Last Name:

E-mail:

Please make sure your email address is correct. A confirmation link will be sent to this address.

Phone:

Role:

Industry:

Company:

Country:

Registration will enable us to provide you with professional services:

- Customer support for all problems, questions, and suggestions.
- Access to early releases and evaluations.
- New versions, patches, and updates.

Please share your experience about the tool (optional):

Register Register Later Go to Web Page

Figure 15 -- The Registration Dialog

Registration Workflow

The registration process is straightforward. After a new key application, you will be requested to register your installation. If you have successfully sent the online registration form, you will receive an email with a link to confirm your registration and the correctness of information furnished through online registration.

Upon confirmation, a dedicated account will be created for you at www.magicdraw.com.

NOTE

If you have an existing profile at www.magicdraw.com you may register with the same user information and the same profile information will be used for registration.

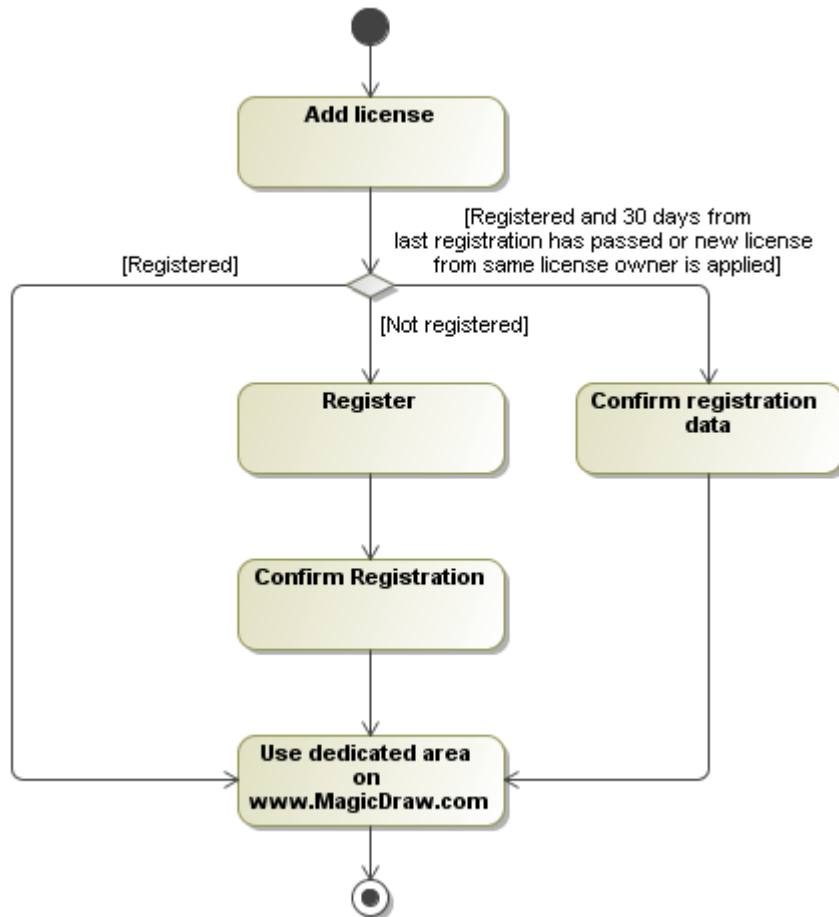


Figure 16 -- Registration Workflow

1. Add a License

MagicDraw always checks the registration status at startup after license has been provided, an unlock key has been added, or a Floating server has been selected.

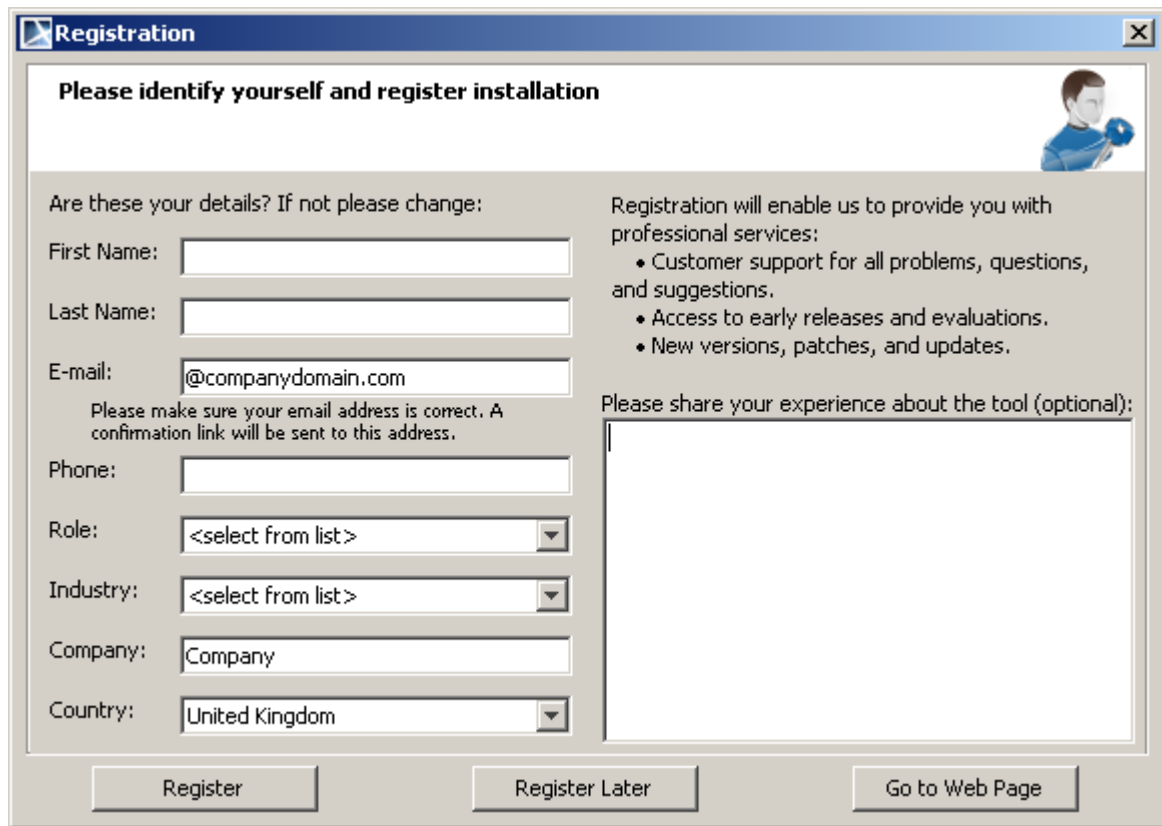
2. Registering

If you are not a registered user, MagicDraw will open the **Registration** dialog at startup, prompting you to register.

NOTE You do not have to complete the online User Registration to use MagicDraw, however, it is recommended to do so to receive the benefits available for a registered user. You can complete Product Registration at any time by clicking **Register...** on the MagicDraw **Help** menu.

To complete Product Registration:

1. On the MagicDraw main menu, click **Help > Register...** .The **Registration** dialog will open (Figure 17 on page 59).
2. Provide the requested information in the **Registration** dialog (some information has been pre-filled from the key owner profile in order to provide a more usable registration process).
3. Share your experience about the tool (optional).
4. Click **Register** to send data to the server and to receive a confirmation email later on.



The image shows a 'Registration' dialog box with a title bar containing a close button. The main heading is 'Please identify yourself and register installation'. On the right, there is a small cartoon illustration of a person with a magnifying glass. The dialog is divided into two main sections. The left section, titled 'Are these your details? If not please change:', contains several input fields: 'First Name:', 'Last Name:', 'E-mail:' (with the text '@companydomain.com' inside), 'Phone:', 'Role:' (a dropdown menu showing '<select from list>'), 'Industry:' (a dropdown menu showing '<select from list>'), 'Company:' (with the text 'Company' inside), and 'Country:' (a dropdown menu showing 'United Kingdom'). Below the 'E-mail:' field, a note states: 'Please make sure your email address is correct. A confirmation link will be sent to this address.' The right section, titled 'Registration will enable us to provide you with professional services:', lists three bullet points: '• Customer support for all problems, questions, and suggestions.', '• Access to early releases and evaluations.', and '• New versions, patches, and updates.' Below this list is a section titled 'Please share your experience about the tool (optional):' followed by a large empty text area. At the bottom of the dialog, there are three buttons: 'Register', 'Register Later', and 'Go to Web Page'.

Registration

Please identify yourself and register installation

Are these your details? If not please change:

First Name:

Last Name:

E-mail:

Please make sure your email address is correct. A confirmation link will be sent to this address.

Phone:

Role:

Industry:

Company:

Country:

Registration will enable us to provide you with professional services:

- Customer support for all problems, questions, and suggestions.
- Access to early releases and evaluations.
- New versions, patches, and updates.

Please share your experience about the tool (optional):

Register Register Later Go to Web Page

Figure 17 -- The Registration Dialog

NOTES

- You can click the **Confirm Later** button to register at a later time and close the **Registration Confirmation** dialog.
- A message reminding about Registration Confirmation will appear at MagicDraw startup if you do not complete the registration process.
- No Magic, Inc. respects your privacy. We will only use your personal information for communications and management of your online account, and the products you register with your account.

3. Confirming Your Registration

An email with registration data and a confirmation link will be sent to the e-mail address provided during registration. Click the confirmation link to confirm the registration process and create or navigate (if it has been created) to your dedicated area at www.magicdraw.com.

4. Logging in to Your Dedicated Area at www.magicdraw.com

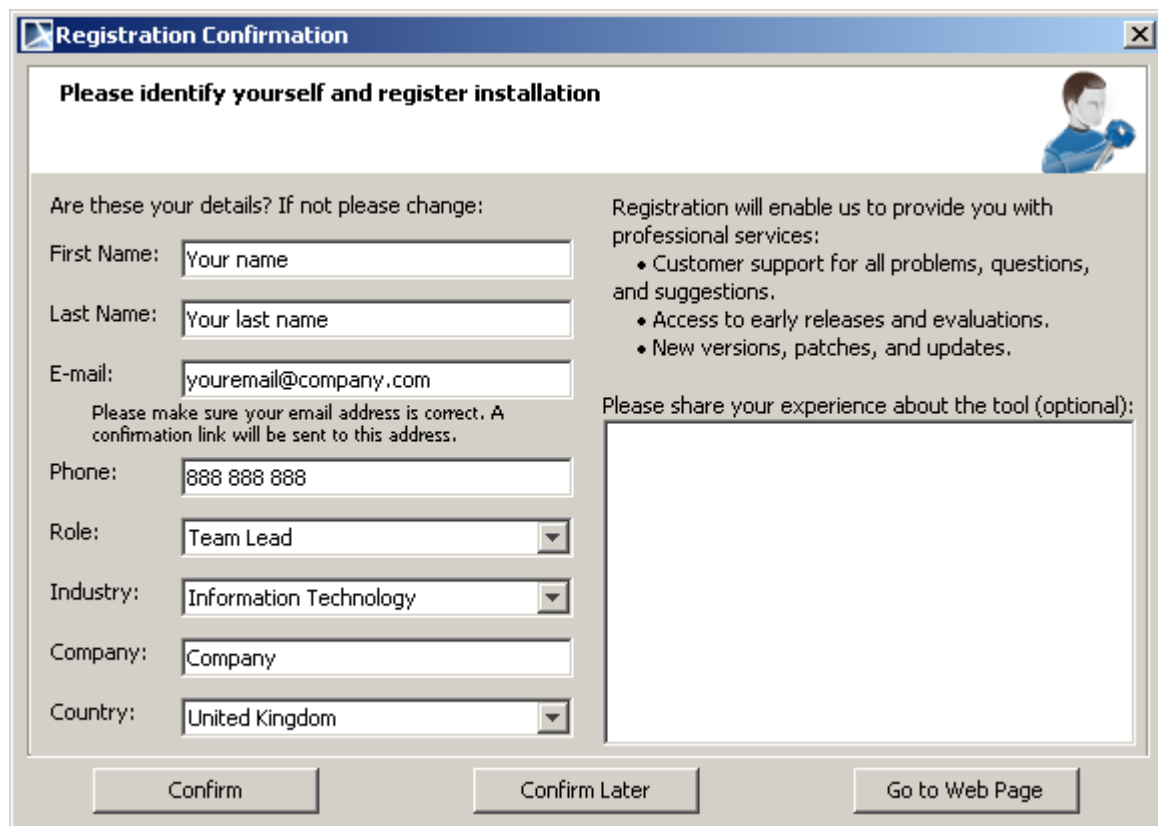
Use your login name and password received upon confirming your registration to log in to the user dedicated area at www.magicdraw.com.

Your registration will enable us to provide you with the following professional services:

- Dedicated customer support for all problems, questions, and suggestions.
- Access to early releases and evaluations.
- New versions, patches, and updates.

Registration Data Confirmation

After a period of 30 days has passed since the date of your registration or a new license from the same licensed owner has been applied, you will be requested to confirm that you are the one using the software installation at MagicDraw startup. A profiled **Registration Confirmation** dialog will open. Check the correctness of data and click the **Confirm** button.



The image shows a 'Registration Confirmation' dialog box with a blue title bar and a close button. The main heading is 'Please identify yourself and register installation'. On the right, there is a small icon of a person with a magnifying glass. The form is divided into two columns. The left column contains a heading 'Are these your details? If not please change:' followed by input fields for 'First Name' (containing 'Your name'), 'Last Name' (containing 'Your last name'), 'E-mail' (containing 'youremail@company.com'), 'Phone' (containing '888 888 888'), 'Role' (a dropdown menu with 'Team Lead' selected), 'Industry' (a dropdown menu with 'Information Technology' selected), 'Company' (containing 'Company'), and 'Country' (a dropdown menu with 'United Kingdom' selected). Below the 'E-mail' field, a note states: 'Please make sure your email address is correct. A confirmation link will be sent to this address.' The right column contains a heading 'Registration will enable us to provide you with professional services:' followed by a bulleted list: '• Customer support for all problems, questions, and suggestions.', '• Access to early releases and evaluations.', and '• New versions, patches, and updates.' Below this is a heading 'Please share your experience about the tool (optional):' followed by a large empty text area. At the bottom, there are three buttons: 'Confirm', 'Confirm Later', and 'Go to Web Page'.

Registration Confirmation

Please identify yourself and register installation

Are these your details? If not please change:

First Name:

Last Name:

E-mail:

Please make sure your email address is correct. A confirmation link will be sent to this address.

Phone:

Role:

Industry:

Company:

Country:

Registration will enable us to provide you with professional services:

- Customer support for all problems, questions, and suggestions.
- Access to early releases and evaluations.
- New versions, patches, and updates.

Please share your experience about the tool (optional):

Figure 18 -- Registration Confirmation

NOTE You will receive no email upon confirming your registration.

If you are registering as a new user, with different profile information, the **Update Existing Profile** question will appear. You can either update your existing profile or identify yourself as a different user using the product installation.

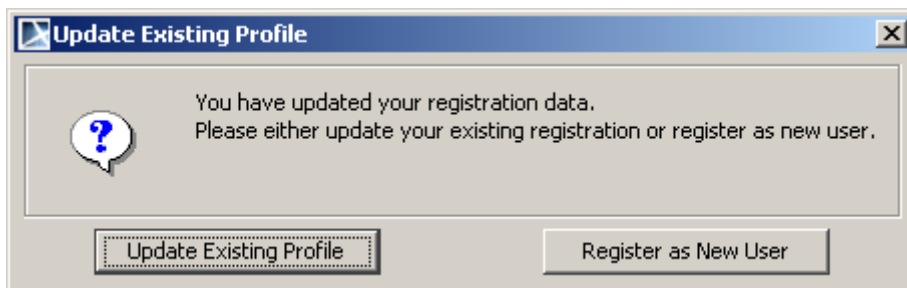


Figure 19 -- Updating Existing Profile

Bug Report

If you are a registered user, your personal information will be provided in the **Submit a Bug** dialog. Submit notifications of software errors dialog is available from **Help** menu > **Submit a Bug...**

If the provided information does not correspond to the information you have provided during registration, once you click the **Send** button to submit the bug, you will be asked to register or update your information. The **Registration** dialog will open and profiled with your personal information from previous registration with changes from the **Submit a Bug** dialog or details from the **Submit a Bug** dialog if you have not yet registered.

For more information about bug reporting, see “Bug Reports” on page 43.

Used Licenses Management

If you are a license owner you can review used licenses and registration under your license from your dedicated area at www.magicdraw.com by clicking the **Used Licenses List** menu (on the right-hand side).

Each installation using your unlock key is associated with your license owner profile.

The used licenses will be shown in table. A single entry will be shown per installation. For columns information refer to Table 1 on page 63.

TABLE 1. Used License List

Column name	Description
Profile ID	The license user's profile ID.
First Name	The license user's first name.
Last Name	The license user's last name.
Email	The license user's email address.
Date	Date of last activity.
Product, Edition, Version	Product, Edition, and Version of a particular installation / registration.
Internal IP / External IP	The Internal and External IP addresses of the machine on which the product is installed / registered.
Installation Status	For Installation status refer to Table 2 on page 64. As a license owner you can change your registration status, except for uninstalled status because it is not editable.

NOTE If installation is not registered less data will be visible in Used Licenses List table.

TABLE 2. Installation status

Status	Description
Not Registered	The product has been installed but registration is not yet performed.
Waiting for Confirmation	The product has been installed, registered, the registration email has been sent, but registration has not yet been confirmed. Note: You can confirm your registration manually by changing its status.
Registered	The product has been installed, registered, and registration has been confirmed.
Illegal	A stolen product license or illegal installation. You may switch installation to this status if you suspect one of the above reasons is the cause. The user using the product installation will receive a message about illegal use of license and will be suggested to contact the license owner. Use of product is restricted after 3 days until a valid key is provided.
Uninstalled	The product has been uninstalled.

Illegal Usa

You may receive an illegal use message due to one of the following reasons:

1. The key is illegal and has no corresponding purchase.

The message will be:

"The key created for <key user name> is invalid. MagicDraw usage will be restricted after 3 days until a valid key is provided.

Please contact your administrator or key owner.

If this is mistake please contact us at registration@nomagic.com, to solve this issue."

2. Your product installation has been identified as illegal by the license owner or license tracker.

The message will be:

2 GETTING STARTED

User Registration

"MagicDraw installation is marked as illegal. You are either not entitled to this key created for < key user name>, or were wrongly identified.

Magic Draw usage will be restricted after 3 days until a valid key is provided.

Please contact your administrator or key owner.

If you are the key owner and suspect wrong identification, please contact us at registration@nomagic.com to solve this issue."

NOTE

If you are the license owner and suspect wrong identification, please contact us at registration@nomagic.com to solve the problem.

Notifications of Registration

A notification of registration will be sent to the license owner during the registration process.

You can turn on or turn off the option for the license owner to receive a notification of registration "Notify key holder on registration" by logging in to your dedicated area at www.magicdraw.com and click the **Used Licenses List** menu (on the right-hand side).

Troubleshooting

Issue	Solution
Registration is requested on each startup	<p>The Registration Confirmation dialog will open each time MagicDraw starts until you register the product installation.</p> <p>The registration process is straightforward and requires minimum data for you to access dedicated online support, answers database, new products evaluation, and beta products.</p> <p>Clicking the Confirm Later button will close the Registration Confirmation dialog.</p>
Registration confirmation is requested at each startup	<p>If you do not click the registration confirmation link, installation will not be confirmed.</p> <p>You may request a new confirmation email from the registration confirmation message.</p>
No email with a confirmation link is received	<p>Your spam or virus filter may filter out the email with a confirmation link. If that is the case, you can request a new confirmation email by clicking Help > Register... on the MagicDraw main menu.</p>
The No connection to registration server message opens	<p>Check your internet connection and try to reconnect.</p> <p>NOTE: User Registration is encouraged, but is not required to use MagicDraw.</p>
You are getting can not connect to server message on registration dialog invocation from Help menu -> Registration	<p>Check your internet connection and try to reconnect.</p> <p>NOTE: User Registration is encouraged, but is not required to use MagicDraw.</p>
An illegal use of key or illegal installation message opens	<p>An illegal use of key or illegal installation status has been detected. For more information refer to section "Illegal Usa" on page 64.</p>
As a license owner you suspect that installation is illegal.	<p>Go to your dedicated area at www.magicdraw.com and click Used Licenses List and switch to the illegal installation status. For more information refer to section "Used Licenses Management" on page 62.</p>

Issue	Solution
The Registration dialog does not opens at Startup	<ul style="list-style-type: none">You have already registered and the period of 30 days since the date of your last registration has not been used up.There are network limitations to check your registration status. <p>Note: User Registration is encouraged, but is not required to use the MagicDraw.</p>
I'm receiving notifications about registrations with my license during team member registration process	<p>A notification of registration will be sent to the license owner during the registration process.</p> <p>You can turn on or turn off the option for the license owner to receive a notification of registration "Notify key holder on registration" by logging in to your dedicated area at www.magicdraw.com and click Used License List.</p>

NOTE

If you encounter problems during the registration process, please contact us registration@nomagic.com

Updating

An automatic updates feature is implemented in MagicDraw. Notification and update of all the patches can be done automatically.

To enable an automatic checking for MagicDraw updates

1. Open the **Environment Options** dialog box, **Update** pane. For a detailed description of this dialog box, see "Update pane" on page 145.
2. Select a period when MagicDraw should check for updates from the **Check for Updates** drop-down list box: **Manually**, **On startup**, **Once a day**, **Once a week**, **Once a month**. It is recommended to check for updates once a month.

NOTE

You may also manually check for updates: go to the **Help** menu and select the **Check for Updates** command.

Auto-Check for Updates dialog box

The dialog box is opened when automatic checking for MagicDraw updates is enabled on the **Update** pane in the [Environment Options](#) dialog box and the **Show Auto-Checking Confirmation Dialog Box** check box is selected.

Element Name	Function
Show this tip next time	If selected, the dialog box is opened each time that MagicDraw should be updated according to the schedule selected on the Update pane in the Environment Options dialog box.
Check	Starts checking for MagicDraw updates in the http://www.magicdraw.com page.
Cancel	Closes the dialog box without saving changes.
Help	Displays MagicDraw Help .

3 USING MAGICDRAW

In “Using MagicDraw”, you will find information about how to define MagicDraw according to perspective, an introduction to the MagicDraw User Interface, and defining your environment.

- "Perspective Selection and Customization", on page 69.
- "MagicDraw User Interface", on page 78.
- "Menu System", on page 80.
- "Toolbars", on page 102.
- "Using the Browser", on page 105.
- "Setting Environment Options", on page 129.
- "Look and Feel: Controlling the Interface", on page 163
- "Assigning Shortcut Keys", on page 165.

Perspective Selection and Customization

Launch MagicDraw for the first time and after the application starts, the **MagicDraw Startup** dialog box appears. In this dialog box you may select your work perspective.

Due to the growing number of MagicDraw features, many features may be configured for standard or expert user. MagicDraw can satisfy the needs of different software development process roles. In order to better satisfy user needs, MagicDraw configuration depends on Perspective.

Perspectives allow:

- The selection of a predefined MagicDraw configuration and features according to your software development process role.
- Finding features faster, because there are less of them.
- To choosing a suitable MagicDraw experience mode with a single click.
- Customizing a set of predefined features and configuration based on user needs.

There are five perspective modes in MagicDraw:

- **Business Analyst** - this perspective provides features for the Business Analyst. This role is responsible for defining business architecture. Code engineering, transformations, extensions, and other features are hidden.
- **Full Featured** - Perspective provides all features available in MagicDraw and installed plugins.
- **Quick Start** - Quick Start perspective provides basic features dedicated for modelling and not overcrowded interface for quick learning. Code engineering, transformations and other advance features are hidden, however easily reachable in expert mode of this perspective.
- **Software Architect** - this perspective provides features primarily involved in designing and implementing projects. It is a set of roles consisting of Software Architect, Designer, Interface designer, and Database designer. This is the default MagicDraw configuration. All functionalities are available for expert mode.
- **System Analyst** - this perspective provides features primarily dedicated to obtaining requirements and modeling the system. Analysis features are highlighted. Configuration is modeling oriented. Code engineering, transformations and other features are hidden.

To set the perspective for the MagicDraw environment

- Launch MagicDraw for the first time. The **MagicDraw Startup** dialog box will appear with the possibility to switch between perspectives. Select the desired perspective from the list and click **OK**.
- From the **Options** main menu, choose **Perspectives** and then **Perspectives**. The **Select Perspective** dialog box opens. Select the desired perspective and click **Apply**.

Customizing MagicDraw Perspectives

Perspective customization allows the grouping of functional MagicDraw features to standard/expert modes. Customization also allows the user to hide unnecessary commands, which makes MagicDraw simpler and faster to use.

To open the **Customize Perspectives** dialog box

- From the **Options** main menu, choose **Perspectives** and then **Customize**.
- In the Perspectives dialog box, click the **Customize** button.

MagicDraw has six customizable areas in different perspectives. Each of these areas has a set of commands, which can be shown in standard/expert mode, or hidden:

- Main menu;

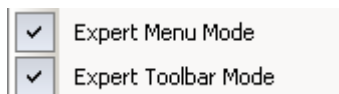
- Main toolbars;
- Diagram toolbar;
- Diagram modeling elements toolbar;
- Context menu actions;
- Reports.

To customize the selected MagicDraw area in a predefined perspective

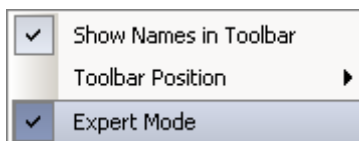
1. In the **Customize Perspectives** dialog box, select the perspective and click the **Edit** button near the selected MagicDraw area. The appropriate **Customize** dialog box opens.
2. Expand tree sections and select radio buttons beside items in the tree depending on your choice for **Standard and Expert**, **Expert** only or **Hidden** modes.
3. Click **OK** to save changes, then **OK** in the **Customize Perspectives** dialog box, and then **Apply** in the **Select Perspective** dialog box (if needed).

To switch between Standard/Expert menu, toolbar, or diagram toolbar modes

- From the toolbar shortcut menu, select/clear the **Expert Menu Mode** or **Expert Toolbar Mode** check box.



- From the diagram buttons toolbar menu, select/clear **Expert Mode** check box.



MagicDraw Startup dialog box

Launch MagicDraw for the first time. The **MagicDraw Startup** dialog box will appear with the possibility to switch between perspectives.

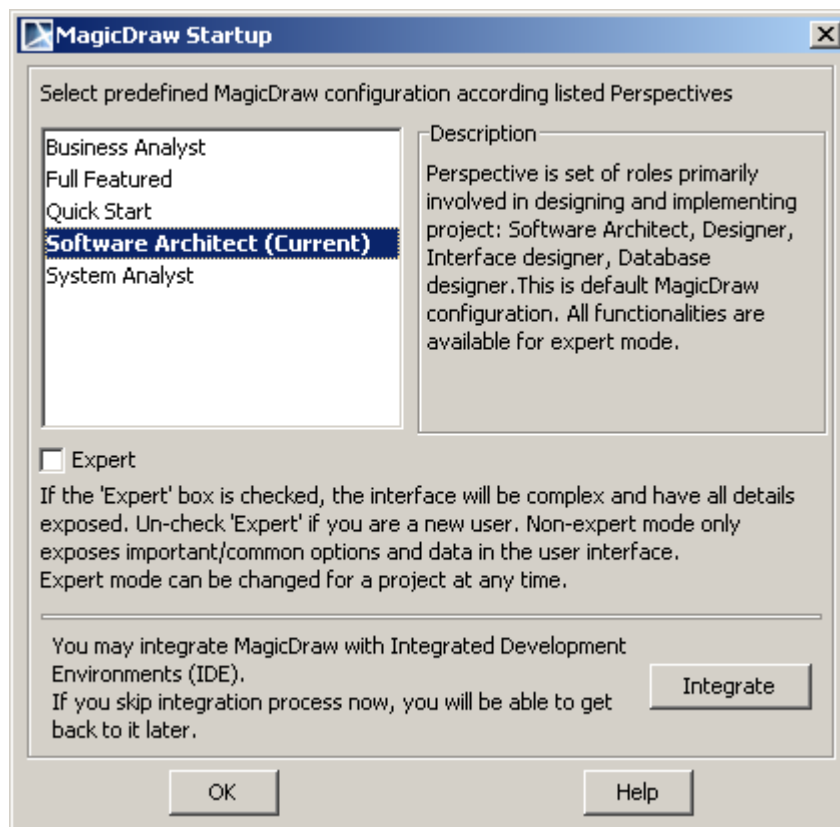


Figure 20 -- MagicDraw Startup dialog box

Element Name	Function
Business Analyst Full Featured Quick Start Software Architect System Analyst	Possible perspectives to set, which will load a predefined MagicDraw configuration.

Element Name	Function
Expert	If not selected, a simplified MagicDraw interface with the most popular items and features will be opened and the specification properties will be shown in Standard Mode.
Integrate	Opens the Integrations dialog box for quick integration with the selected tool.
OK	Loads MagicDraw with the selected perspective.
Help	Displays MagicDraw Help .

Select Perspective dialog box

From the **Options** main menu, choose **Perspectives** and then **Perspectives**.

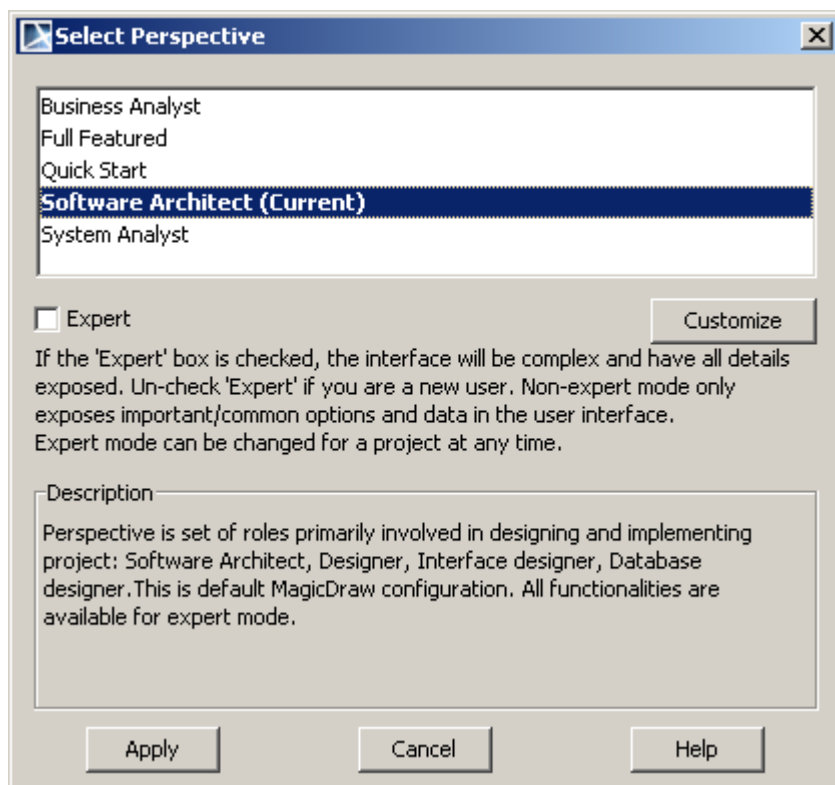


Figure 21 -- Select Perspective dialog box

Element Name	Function
Business Analyst Full Featured Quick Start Software Architect System Analyst	List of possible perspectives, which will load a predefined MagicDraw configuration.
Customize	Opens the Customize Perspective dialog box.
Expert	If not selected, a simplified MagicDraw interface with the most popular items and features will be opened and the specification properties will be shown in Standard Mode.

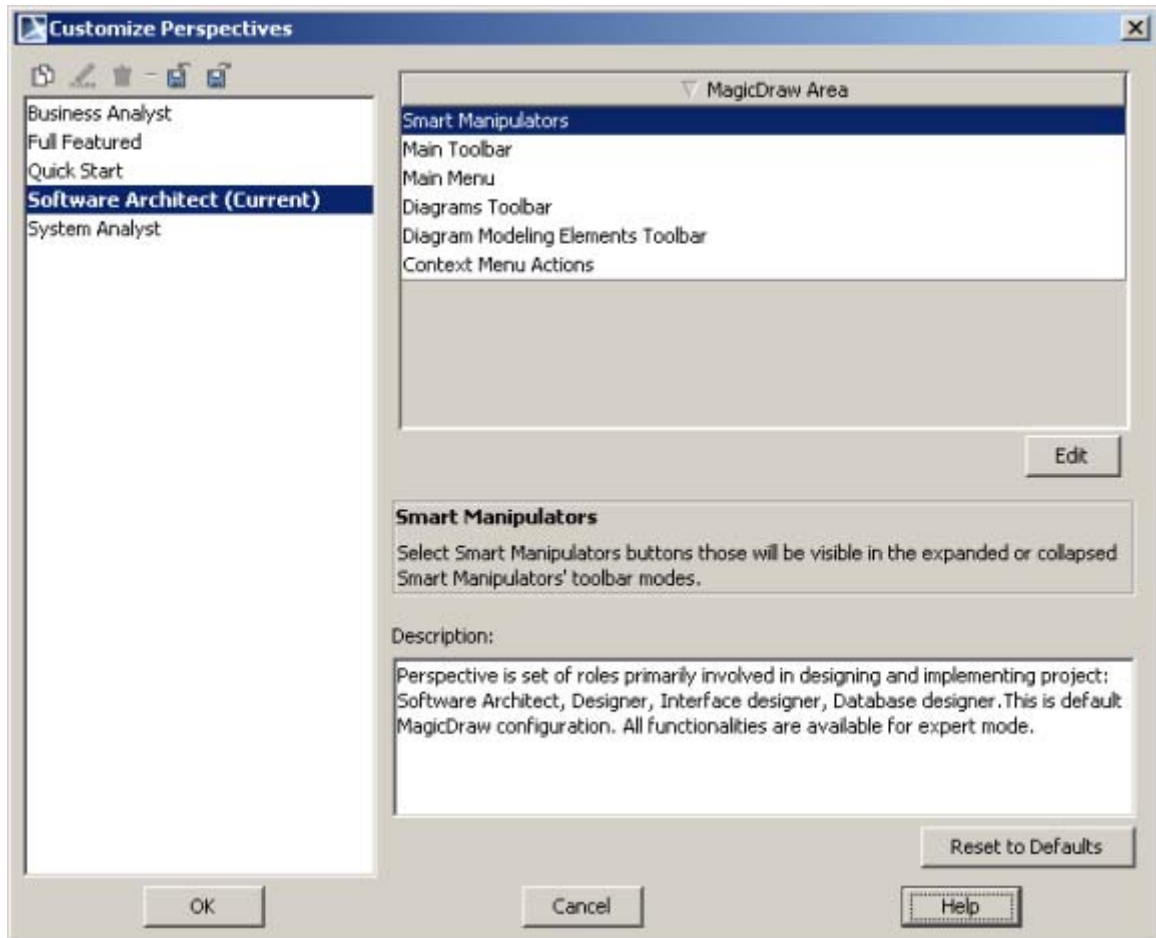
3 USING MAGICDRAW

Perspective Selection and Customization

Element Name	Function
Apply	The selected perspective will be applied for current MagicDraw mode.
Cancel	Closes the dialog box without saving changes.
Help	Displays MagicDraw Help .

Customize Perspectives dialog box






From the **Options** main menu, choose **Perspectives** and then **Customize**.



3 USING MAGICDRAW

Perspective Selection and Customization

Figure 22 -- Customize Perspectives dialog box

Element Name	Function
Clone Selected Perspective 	Copies the selected perspective to a new one.
Rename Selected Perspective 	The Enter Perspective Name dialog box opens. Change name of the perspective and click OK . Renaming can also be performed using the F2 shortcut key.
Remove Selected Perspective 	Deletes the selected perspective from the list.
Import New Perspective 	The Open dialog box appears. Select *.umd extension file and click Open to import the perspective into the MagicDraw environment.
Export Selected Perspective 	The Save dialog box opens. Type a name for the created perspective and click Save to store it as *.umd extension file.
Business Analyst Software Architect System Analyst	List of possible perspectives, which will load the predefined MagicDraw configuration.
MagicDraw Area	List of customizable toolbars and command sections.
Edit	Click the Edit button to open the Customize Main Menu dialog box in which a commands mode could be changed by selecting radio buttons.
Description	Displays short description about each selected area.
Reset to Defaults	Resets changes back to the default configuration.

MagicDraw User Interface

MagicDraw main window has the following main parts:

- main menu
- main toolbar
- browser
- diagram toolbar
- diagram pane

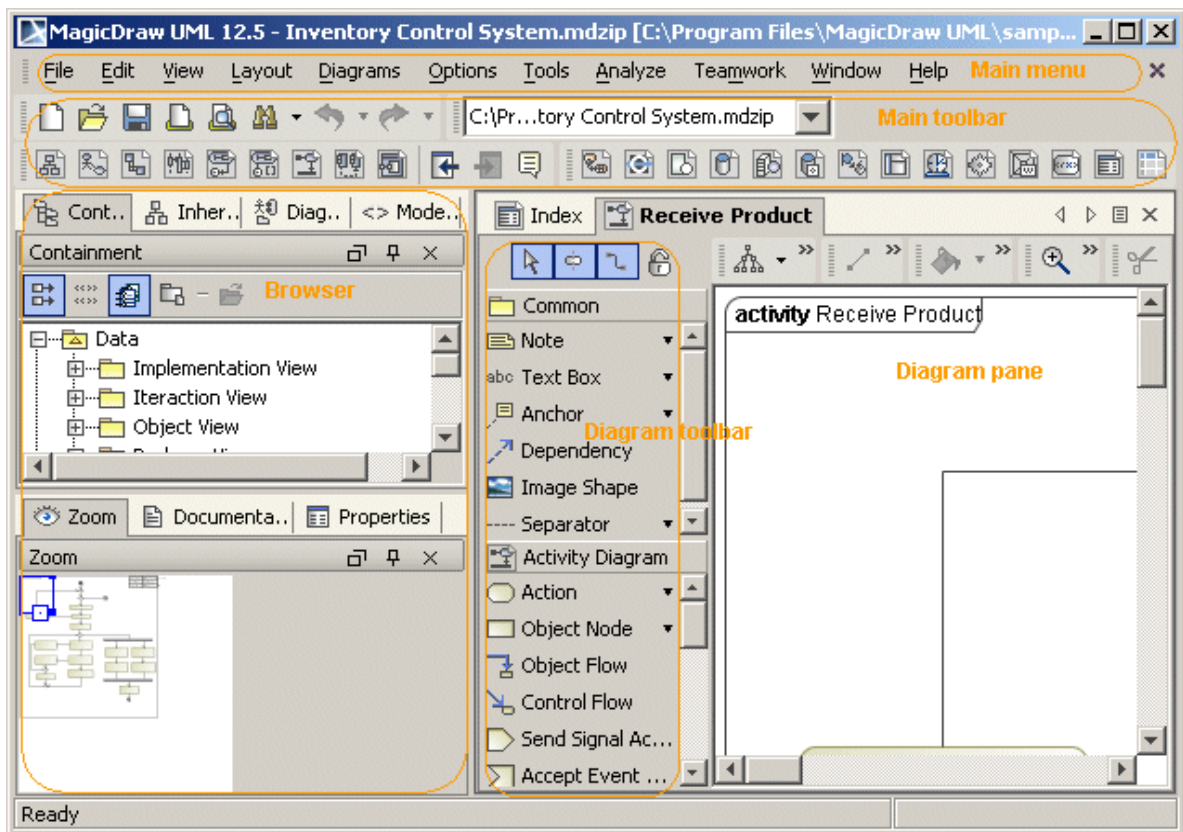


Figure 23 -- MagicDraw main UI structure

Nearly all MagicDraw commands can be accessed from multiple places within MagicDraw;

- Main menu
- Main toolbar
- Browser
- Diagram toolbar
- Shortcut menus (accessible by right-clicking)
- Shortcut keys
- Smart manipulators (accessible by selecting a symbol on the diagram pane).

The table below shows how several commands are accessible in various ways:

Function	Accessibility
Main operations of editing (copy, cut, paste, delete)	<ul style="list-style-type: none"> • Edit menu. • Main toolbar buttons. • Shortcut keys. • Shortcut menu commands from Browser.
Opening of Specification dialog boxes	<ul style="list-style-type: none"> • By double-clicking the model element. • Specification command from the element shortcut menu on the diagram or Browser. • When adding one model element to another model element from the Specification dialog box or Browser.
Defining symbols properties (font, color, etc.)	<ul style="list-style-type: none"> • Symbol shortcut menu -> Symbol(s) Properties. • Options menu -> Project. • Main toolbar buttons




NOTES

The Symbol shortcut menu is also accessible from the **Edit** menu, **Symbol**.



The toolbar of a particular diagram presents the paths and shapes available for the corresponding diagram. If an arrow is placed on the diagram toolbar button, select a button representing the corresponding model element by right-clicking the button.

Menu System

File menu

Command	Button / Hot keys	Function
New Project	 CTRL+N	<p>Creates a new project. Choose an icon to create a new Blank Project, create a Project from an Existing Source, or create a Project from a Template.</p> <p>A project is nameless until you close or save it by choosing the Save Project or Save Project As commands.</p> <p>You may simultaneously create as many new projects as you wish, without saving and closing the previously created or open projects. You may work on only one open or created project at a time. If you wish to work with another project, select the project name in the Projects list on the main window toolbar.</p>
Open Project	 CTRL+O	<p>Opens an existing project.</p> <p>The Open dialog box appears. Select a project you wish to open. You may open as many projects as you wish. If you wish to work with another project, select the project name in the Projects list on the main window toolbar.</p>
Save Project	 CTRL+S	<p>Saves the open project.</p> <p>To save the current project for the first time, type the name of the project and select the format of the file you wish to save.</p>
Save Project As		<p>Saves the project giving it a name.</p> <p>Use this command to save the current project for the first time or under a different name, the Save dialog box appears. Type the name of the project and select the format of the file you wish to save.</p>



Command	Button / Hot keys	Function
Close Project		Closes an open project. If the open project has unsaved changes, MagicDraw displays a dialog box asking whether the changes should be saved before the project is closed.
Close All Projects		Closes all open projects. If the open projects have unsaved changes, MagicDraw displays a dialog box asking whether the changes should be saved before the each project is closed.
Open Element from URL		You can open any elements through their URLs by clicking the Open Element from URL command and the element will be highlighted in the Containment tree or in the diagram. For more information about element URL, see “Copying/Opening Element URLs” on page 385.
Import MagicDraw Project		The Import dialog box appears. Select the project you wish to import. You may import as many projects as you wish. Imports an existing project to a previously open project. This is the recommended command for importing an existing project to the teamwork server.
Use Module		The Use Module dialog box appears. Choose profile or module for use in the project.
Export		<ul style="list-style-type: none"> • Module - saves the content of the selected objects in a separate file. • Template - saves the current project as a template project. • EMF UML2 (v1.x) XMI - exports the MagicDraw model to EMF based UML 2 compatible XMI. • BPEL - exports the Business Process diagram to the Business Process Execution Language (BPEL), based on BPEL specification v.1.1
Share Packages		Saves the package as a separate module.




Command	Button / Hot keys	Function
Save As Image		Saves the open diagram as an image of in the selected formats. The Save As Image command is used for saving any item or group of items selected in the diagram pane, in one of the various image file formats.
Print	 CTRL+P	The Print dialog box appears. Printing of active diagram, selected symbols, or selected diagrams is available. Select the printer, set the options for the printer, specify the number of copies, and select the specific pages to be printed.
Print Preview		Preview the diagram before printing. The Print Preview screen opens.
Print Options		The Print Options dialog box opens.
Project Properties		The Project Properties dialog box opens. This dialog box contains the following information: project location, file size, date created, date modified for the last time, and number of diagrams in the project. In the Description tab, type the description of the project or other important information. The Modules tab shows a list of modules the project is using. The list shows a specified number of recent project files. Specify the number of files in the Recent Files List Size property in the Environment Options dialog box.
Opened Projects		A list of open projects.
Exit		Exits the application. All open projects are closed. If an open project has unsaved changes, the MagicDraw displays a dialog box asking whether the changes should be saved before closing the project.


The **File** menu contains a list of recent projects. The shortcuts with numbers to recent projects are displayed. If the shortcut is selected from the **File** menu, the recent project will open instantly.


Edit menu

MagicDraw allows you to use the **Edit** menu commands while drawing the diagrams on the Diagram pane. The commands allow selecting, cutting, copying, and pasting of items and entire diagrams, reversing the actions you have taken while drawing, and finding an item in the current project.




Command	Button/ Shortcut keys	Function
Undo	 CTRL+Z	<p>Reverses the last action you have performed while drawing the diagram on the Diagram pane (moving, dragging, resizing, cutting, copying, pasting, deleting, selecting, editing shapes, setting project and shape properties, etc.). Actions are reversed in the opposite order you have performed them, starting with the most recent.</p> <p>By default, the limit of the undo mechanism is 100 steps backwards.</p> <p>To change the limit, choose Environment from the Options menu. The Environment Options dialog box appears. Change the Undo List Size property.</p> <p>The Undo command is unavailable until you perform any action after loading an the existing project or creating a new project.</p> <p>Each command has an easily recognized name. You will be able to see the command history and undo or redo one action or a group of actions. The main window will have two lists of commands: one for the undo commands, another one for the redo commands.</p>
Redo	 CTRL+Y	<p>Reverses the action of the Undo command (moving, dragging, resizing, cutting, copying, pasting, deleting, selecting, etc.). The Redo command is unavailable until you use the Undo command.</p>

Command	Button/ Shortcut keys	Function
Cut	 CTRL+X	<p>Cuts the selected items or group of items on the Diagram pane.</p> <p>The cut items are placed in the clipboard. Later they can be pasted back to the Diagram pane of the current or to another project.</p> <p>The Cut command is unavailable until you select any item or any group of items on the Diagram pane of the current project.</p>
Copy	 CTRL+C	<p>Copies the selected items or group of items on the Diagram pane.</p> <p>The copied items are placed in the clipboard. The cut items can be pasted back to the Diagram pane or to another project.</p> <p>The Copy command is unavailable until you select any item or any group of items on the Diagram pane of the current project.</p>
Copy URL		<p>Copy a project element URL to a clipboard and share it with other as a quick reference to model elements. For more information about copying element URL, see “Copying/Opening Element URLs” on page 385.</p>
Paste	 CTRL+V	<p>Pastes the cut or copied items or group of items from the clipboard to the Diagram pane of the current project.</p> <p>MagicDraw creates shapes for items, or a group of items, in the current project. You will see the data and shapes of the pasted items and diagrams in the Browser window.</p>
Paste With New Data	CTRL+E	<p>Pastes the cut or copied items or groups of items from the Clipboard to the Diagram pane of the current project.</p> <p>MagicDraw creates new data and data shapes in the current project.</p>

Command	Button/ Shortcut keys	Function
Delete	 CTRL+D	Deletes data together with symbol. It is unavailable until you select any item or group of items in the current project.
Delete Symbol(s)	DELETE	Deletes a symbol of model element from the diagrams, by leaving it in the model.
Select All	CTRL+A	Selects all items on the Diagram pane of a particular project.
Select All of the Same Type	CTRL+ALT+A	Selects all shapes of the selected types in active diagram. Enabled, when one or more shapes are selected in active diagram.
Copy as BPM	CTRL+SHIFT+B	Copies the selected model elements to the system clipboard. If no model elements are selected, the active diagram is copied.
Copy as EMF	CTRL+SHIFT+E	<p>Copies the selected model elements to the system clipboard. If no model elements are selected, the active diagram is copied.</p> <p>NOTE Copying as EMF is available only under Windows system.</p>
Copy as JPG	CTRL+SHIFT+J	<p>Copies the selected model elements to the system clipboard as a JPG image. If no model elements are selected, the active diagram is copied.</p> <p>NOTE Copying as JPG is available only under Windows system.</p>
Copy as PNG	CTRL+SHIFT+P	<p>Copies the selected model elements to the system clipboard as a PNG image. If no model elements are selected, the active diagram is copied.</p> <p>NOTE Copying as PNG is available only under Windows system.</p>

Command	Button/ Shortcut keys	Function
Find	 CTRL+F	Opens the Find dialog box.
Quick Find	CTRL+ALT+F	Performs a quick search of class/interface, classifier, or diagram.
Find TODO		Performs a search for the TODO tagged value. Results are displayed in the Browser, Search Results tab.
Paths		<ul style="list-style-type: none"> • Path Style - choose the line style for drawing a path. • Rectilinear – drawing rectilinear lines. • Oblique - drawing free form lines. • Bezier - in computer graphics, a curve that is calculated mathematically to connect separate points in smooth, free-form curves. • Change Path Style – switches in series between rectilinear, oblique, and bezier path line style. Shortcut is CTRL+L. • Reset Labels Positions – resets the changed path labels (name, roles, etc.) to the default position. • Remove Break Points – removes the break points of the path and makes the path a line straight.
Symbol		All commands that are available through the shortcut menu for a particular symbol.

View menu

Command	Button / Hot keys	Function
Fit In Window	CTRL+W	Reduces the size of the whole diagram to fit in the Diagram pane.
Zoom In	 CTRL+NUMPAD PLUS SIGN (+)	To change the zoom step size choose Environment from the Options menu and set the Zoom Step Size property in the Environment Options dialog box.
Zoom Out	 CTRL+NUMPAD MINUS SIGN (-)	Decreases the size of the selected objects in the diagram pane by x percent. To change the zoom step size, choose Environment from the Options menu and set the Zoom Step Size property in the Environment Options dialog box.
Zoom 1:1	 CTRL+NUMPAD SLASH MARK (/)	Restores the original size of the selected diagram symbols.
Zoom To Selection	CTRL+NUMPAD ASTERICS MARK (*)	Increases the size of the selected model element on the Diagram pane to the maximum visible size.
Refresh	CTRL+R	Repaints all diagram shapes.

Command	Button / Hot keys	Function
Grid		<p>Set grid options. Every diagram may have its own grid settings:</p> <ul style="list-style-type: none"> • Show Grid - show/hide grid • Snap Paths to Grid - use/do not use grid for drawing paths. • Snap Shapes to Grid - use/do not use grid for drawing shapes. • Grid Size - change the grid size. Type the number.
Recently Closed Diagrams	F12	Opens a list of diagrams that were last closed.
Main Toolbars: <ul style="list-style-type: none"> • Menu Bar • File • Diagrams • Custom Diagrams • Opened Projects • Teamwork • Rearrangible • Hidable • Floatable • Expert Menu Mode • Expert Toolbar Mode • Customize 		Clear the check boxes of the toolbars you want to hide or select/clear check boxes to rearrange toolbar modes.

Command	Button / Hot keys	Function
Diagram Toolbars: <ul style="list-style-type: none"> • Symbol Editing • Path Editing • Edit • View • Layout • Rearrangible • Hidable • Floatable • Expert Mode • Customize 		Clear the check boxes of the toolbars you want to hide or select/clear check boxes to rearrange toolbar modes.
Status Line		Select the check box of Show Status Bar or Show Memory Monitor to display this info on the status line.

Layout menu



Use the commands of the **Layout** menu for managing the layout of the shapes on the current Diagram pane. You must select more than one shape for other Layout menu commands. A class diagram should be open and activate before using the **Class Diagram** command in this menu.

Command	Function
Layout Options	Opens the Diagram Layout Options dialog box. Layout options for the diagram can be set.
Quick Diagram Layout (CTRL+Q)	Applies recommended layout with default options on the active diagram.
Layout Class Diagram Style	Applies layout, which uses specific layout algorithms to improve class diagram readability.
Layout Activity Diagram Style	Applies layout, which uses specific layout algorithms to improve activity diagram readability.
Layout Business Process Diagram Style	Applies layout, which uses specific layout algorithms to improve business process diagram readability.

Command	Function
Layout Hierarchic Style	Applies layout that portrays the main direction or flow of directed graphs. It is ideal for many application areas, especially for Workflow, Software Engineering, Customer relationship management, Configuration management, Process modeling, Database Modeling, and Bio informatics.
Layout Tree Style	Applies layout, which specializes in the layout of tree-structured graphs. The need to visualize directed or undirected trees arises in many application areas, e.g. Dataflow analysis, Software Engineering, Network management, Bio informatics.
Layout Orthogonal Style	Applies layout that is well suited for medium sized sparse diagrams. It produces compact drawings with no overlapping shapes, few crossings, and few bends. All edges will be routed in an orthogonal style, i.e. only rectilinear style links will be used.
Layout Organic Style	Applies layout of the organic style graph.
Layout Circular Style	Applies layout of the algorithm that portrays interconnected ring and star topologies. It is excellent for applications in social networking (criminology, economics, ...), network management, WWW visualization, and eCommerce.
Route Paths Orthogonal Style	Applies layout, which routes the links of a diagram using only vertical and horizontal line segments, while keeping the positions of the shapes in the diagram fixed. The routed links will usually not cross through any shapes and not overlap any other links.
Route Paths Organic Style	Applies layout, which routes the links of a diagram using oblique link style, while keeping the positions of the shapes in the diagram fixed. The routed links will usually not cross through any shapes and not overlap any other links.
Make Same Width	Applies layout to the selected shapes according to their width. After the layout, the width of the shapes is equal (according to the widest).

Command	Function
Make Same Height	Applies layout to the selected shapes according the their height. After the layout, height of the shapes is equal (according to the tallest).
Align <ul style="list-style-type: none">• Right• Left• Top• Bottom	Aligns the selected shapes: <ul style="list-style-type: none">• Aligns the selected shape(s) vertically, starting with the rightmost shape(s).• Aligns the selected shape(s) vertically, starting with the leftmost shape(s).• Aligns the selected shape(s) across from the uppermost shape(s).• Aligns the selected shape(s) across from the lowermost shape(s).
Center <ul style="list-style-type: none">• Horizontally• Vertically	Centers the selected shapes: <ul style="list-style-type: none">• Centers the selected shape(s) on a horizontal line.• Centers the selected shape(s) on a vertical line.
Space Evenly <ul style="list-style-type: none">• Horizontally• Vertically	Sets spaces among the selected shapes evenly. <ul style="list-style-type: none">• Spaces between the selected shapes becomes even on a horizontal line.• Spaces between the selected shapes becomes even on a vertical line.

Diagrams menu

Command	Hot keys	Function
Diagrams		Select the required diagram in the list, to create it.
Customize		The Customize Diagrams dialog box opens.
Diagram Wizards		<p>Wizards for creating diagrams may be opened:</p> <ul style="list-style-type: none"> • Class Diagram Wizard • Package Dependency Diagram Wizard • Package Overview Diagram Wizard • Activity Decomposition Hierarchy Wizard • Hierarchy Diagram Wizard • Realization Diagram Wizard • Sequence Diagram from Java Source Wizard • Content Diagram Wizard
Previous Diagram	 CTRL+0	Activates the previously open diagram.
Next Diagram	 CTRL+9	Activates the next diagram.
Load All Diagrams		If there are unloaded diagrams in the project, this command loads all diagrams.

Options menu

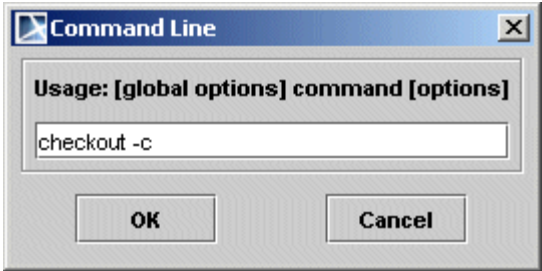
Command	Function
Project	The Project Options dialog box opens.
Modules	The Modules dialog box opens.
Environment	The Environment Options dialog box opens.
Perspectives	Choose a command from the submenu - to switch to different Perspective or to Customize .
Look and Feel	Changes MagicDraw graphic user interface style. Styles: Metal, CDE/Motif, Vsnet (Windows XP), Office 2003 (Windows XP), Eclipse (Windows), Xerto. The Look and Feel Themes command allows you to personalize the graphical user interface: set your favorite colors and fonts.
Interface Style	Single Window , or Multiple Windows interface style.

Tools menu

Command	Hot key	Function
Apply Pattern		The Pattern Wizard dialog box appears. Create the design pattern for the selected class, interface.
Model Transformations		Opens the Model Transformations Wizard dialog box with a list of all available transformations.
Hyperlinks		Opens the Hyperlink dialog where you can add hyperlinks to any model elements.

Command	Hot key	Function
Report Wizard		<p>The Report Wizard is the new report engine for MD 14.0 and above. It is designed to solve the several problems of the old engines (XSL/XSLT and JPython).</p> <p>It supports text based templates to generate the output file. The format of output file depends on the type of the template file. The type of template files that the Report Wizard supports are normal text, RTF, HTML, Spreadsheet template (need to be saved as HTML format), and XML template (DocBook or FO).</p> <p>All commercial MagicDraw editions will have full use of all features within the Report Wizard. MagicDraw Community edition will allow the user to generate output with watermarks.</p> <p>For more details, see the “<i>MagicDraw ReportWizard UserGuide.pdf</i>”, which is located in the MagicDraw installation directory, Manual folder.</p>
Software Design Model Enrichment		Opens the Software Design Model Wizard dialog box.
Use Case Model Enrichment		Opens the Use Case Model Wizard dialog box.
Quick Reverse		Choose the language you need (Java, Java Bytecode, C++, C#, CIL, CIL Disassembler, IDL, CORBA IDL, DDL, EJB, EJB 2.0, XML Schema, WSDL). Opens the Round Trip Set dialog box. (For more details, see Code Engineering User’s Guide).
Generate Code Framework	CTRL+G	Generates code for the selected items in the current diagram. Opens the Message Window with the information appears (For more details, see Code Engineering User’s Guide).
Check Syntax	CTRL+T	Checks syntax in the model according to the default code engineering language. Opens the Message Window with the information.

Command	Hot key	Function
Set empty tags to defaults		<p>Set default tag value to tag with empty value. This functionality is needed when the stereotype is already assigned to an element and the new mandatory tag definition with default value is created for the stereotype. After creating such a tag definition, the model elements that have the modified stereotype applied will have newly created tags unset.</p> <p>For more information see “To create default tag values” on page 814.</p>
Integrations		Opens the Integrations dialog box with a list of tools for possible integration with MagicDraw.
CaliberRM <ul style="list-style-type: none"> • Login • Logout • Launch CaliberRM Client • CaliberRM Requirements 	CTRL+SHIFT+R	<p>Integration with CaliberRM.</p> <p>Opens the Login dialog box for login to CaliberRM server.</p> <p>Logs out from CaliberRM server.</p> <p>Launches CaliberRM Client application.</p> <p>Opens the CaliberRM Requirements window in MagicDraw.</p>

Command	Hot key	Function
CVS <ul style="list-style-type: none"> Command Line 		<p>Performs operations with CVS (for detailed description of integration with CVS, see MagicDraw Integrations User's Guide)</p> <ul style="list-style-type: none"> Allows the user to enter a CVS command line (like "checkout -c") whenever the CVS command is not available through the menus. The Command Line dialog box opens.  <p><i>Figure 24 -- Command Line dialog box</i></p> <ul style="list-style-type: none"> Use this option to checkout a new module on your disk. The Checkout Module dialog box opens. Adds a new project to CVS. The Add Project to CVS dialog box opens. <p>NOTE: You can add, update or commit projects to CVS only if they are saved in some checked out directory or subdirectory.</p> <ul style="list-style-type: none"> Commits a project to CVS. The Commit Project to CVS dialog box opens. Updates the current project by loading the latest project version from CVS.
<ul style="list-style-type: none"> Checkout Module Add Project to CVS Commit Project to CVS Update CVS Project 		

Command	Hot key	Function
ProActivity <ul style="list-style-type: none"> • Import • Export 		<p>The Import ProActivity Files dialog box opens. Specify ProActivity files to start the import process. The UML diagrams will be generated depending on the input files.</p> <p>The Export to ProActivity wizard guides the MagicDraw model export process to ProActivity.</p>
Update C# Language Properties and Profiles		<p>For more information about C# code engineering, read <i>MagicDraw Code&DatabaseEngineering UserGuide.pdf</i></p>
Update C++ Language Properties and Profiles		<p>For more information about C++ code engineering, read <i>MagicDraw Code&DatabaseEngineering UserGuide.pdf</i></p>

Analyze menu

Command	Hot key	Function
Model Visualizer		Opens the Model Visualizer dialog box with a list of all available wizards.
Metrics		Metrics functionality allows the measurement of a project by different viewpoints. Specify the metrics scope in the Metrics Options dialog box.
Compare Projects		Opens the Compare Projects dialog box, where you can choose projects to perform model differencing.

Command	Hot key	Function
Dependency Matrix		<p>Dependency Matrix is a method of visualizing and representing dependency criteria.</p> <p>Diagrams, UML, and extended UML elements serve as row and column entries. The cells in the matrix show where these elements are associated - related.</p> <p>Choose the Create Blank Matrix command to open the Matrix Dependency View</p> <p>Choose the Matrix Templates command to open the Dependency Matrix Templates dialog box.</p>
Validation		For more information about validation, read <i>Validation</i> section.
Validate Shape Ownership		Validates if the symbol owner on the diagram matches the actual element owner in the model. Using the symbol ownership validation feature, shape ownership problems will be highlighted on the diagram pane and you will be able to easily see and resolve them. For more information about shapes checking on diagram see “Resource Manager” on page 497.
Validate Shape Ownership		Validates if a symbol owner on the diagram matches the actual element owner in the model. Using the symbol ownership validation feature, shape ownership problems will be highlighted on the diagram pane and you will be able to easily see and resolve them. For more information see “Resource Manager” on page 497.
Display Paths		Displays paths among shapes that are already created in the model data.
Display Related Elements		Displays elements related to the selected element.








Command	Hot key	Function
Used By	CTRL+ALT+U	Finds a list of all elements that reference the current element.
Depends On	CTRL+ALT+D	Finds a list of the elements that depend on the current element.
Go To		This is a feature that allows you to find model elements associated with the selected model element. Enabled when a selected element is related to another model element.



Teamwork menu

For a detailed description about Teamwork Server, see [MagicDraw Teamwork System User's Guide](#).

Window menu

You may use commands of the **Window** menu to manage the layout of the windows.

Command	Hot key	Function
 Containment		Opens the Containment Tree tab in the Browser window.
 Inheritance		Opens the Inheritance Tree tab in the Browser window.
 Diagrams		Opens the Diagrams Tree tab in the Browser window.
 Model Extensions		Opens the Model Extensions Tree tab in the Browser window.
 Search Results		Opens the Search Results tab in the Browser window.
 Documentation		Opens the Documentation tab in the Browser window.
 Zoom		Opens the Zoom tab in the Browser window

Command	Hot key	Function
 Properties		Opens the Properties tab in the Browser window
 Messages Window	CTRL+M	Opens the Messages Window . The Messages Window is used for displaying the warnings and errors that may appear in the project. It appears automatically and is intended to display the warnings and errors after saving, loading, exporting and importing the project.
Reset Windows Configuration		All Browser tabs are placed in their default position.
List of opened diagrams		Displays a list of open diagrams.
Close All Diagrams But Current	CTRL+SHIFT+F4	Closes all open diagrams except the one you currently being used.
Close All Diagrams	CTRL+ALT+F4	Closes all open diagrams.

The **Window** menu contains a list of open diagrams in the project. The list shows the specified number of the recent diagrams. This number can be customized in the **Recent Windows List Size** property in the **Environment Options** dialog box. For a detailed description on this dialog box, see the Section “Setting Environment Options” on page 129.

Help menu

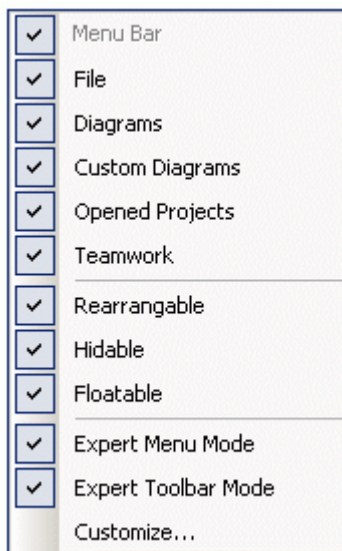
Command	Function
Help Contents	Displays a table of contents for the MagicDraw Help.
User Manual	Opens the MagicDraw User Manual.
Tip of the Day	Displays the Tip of the Day screen.
View and Submit Internal Errors	View errors received during work with MagicDraw. Send a bug report to the MagicDraw support team. For more information about submitting a bug see "Bug Reports" on page 43.
Submit a Bug	The MagicDraw team always welcomes your initiative. Submit bugs, suggestions, and new feature requests in the Submit a Bug dialog box.
Check for Updates	Opens The HTTP Proxy Server Connection dialog box. Set the data for connection to start the MagicDraw UML update.
Unlock MagicDraw/Plugin Key	Change/enter the unlock key in the MagicDraw Unlock Key dialog box.
Resource/Plugin Manager	Check for available updates and new resources in the Resource/Plugin Manager window.
MagicDraw on the Web <ul style="list-style-type: none"> • Online Support • Online Demo • New and Noteworthy • MagicDraw Home Page • UML Stuff • No Magic Home Page 	The WWW pages where you can find additional information about MagicDraw and UML. Get online support.
Entertainment with UML	<ul style="list-style-type: none"> • Memory game. Open pictures to find pairs with as few tries as possible. • Puzzle game. Transpose the separate parts to get the whole picture.
About MagicDraw	Displays the information screen about the MagicDraw tool.

Toolbars

In MagicDraw, you can hide different toolbar parts: right-click the toolbar and select/clear toolbar check boxes you want to be displayed/hidden. You may also save your own toolbar configuration and set it as a default one.

Customizing toolbars

Toolbars configuration shortcut menu has the following commands:



Check box	Function
Rearrangible	If selected, it is possible to change the toolbar position by selecting the dotted line in front of the desired toolbar group and dragging it to a new location.
Hidable	If selected, there is no possibility to close a separately opened toolbar group (for example, dragged diagram pane) with the X button on the right top corner.

Check box	Function
Floatable	If selected, the toolbar group can be dragged to any desirable position inside the MagicDraw borders.
Expert Menu Mode	If selected, all menu commands will be listed on the menu. Otherwise, the command list will be shortened and you can expand it by clicking the arrow on the bottom.
Expert Toolbar Mode	If selected, displays all toolbar buttons, which were marked to be shown in the Expert mode perspective.
Customizet	Opens the Customize Main Toolbars dialog box.

To customize toolbars

1. From the toolbars configuration shortcut menu, select **Customize**. The **Customize Main Toolbars** dialog box opens.
2. Click the **Add** button to append a new button or toolbar to the selected main toolbars section.
3. If add button was selected, the **Add Button** dialog box opens. Choose the desired command from the tree and click **OK**. The button will be added in the selected main toolbar section.

- If the choice was made to add a toolbar, the **Enter Toolbar Name** dialog box opens. Type the name and click **OK** to create a new toolbar section, to which a new button group can be added.

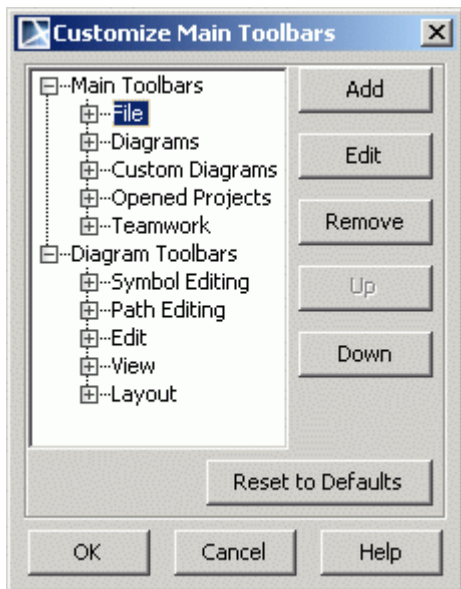


Figure 25 -- The Customize Main Toolbars dialog box

Button	Function
Add	Adds new button or toolbar.
Edit	The Edit Icon dialog box opens. Click the “...” button to add an icon to the selected toolbar button.
Remove	Removes the selected button from the toolbar section.
Up	Moves the selected button up the toolbar list.
Down	Moves the selected button down the toolbar list.
Reset to Defaults	Resets changes made to the toolbar back to the default settings.

Main Toolbar

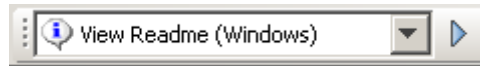
NOTE

To scroll diagram items in the toolbar, select the scrollable toolbar buttons on the diagram toolbar. Using these buttons, you may scroll down or up through the toolbar in order to select the item of the diagram.

The Main Toolbar contains a Projects list box with all open and created project names. MagicDraw UML supports multiple projects, so you may open and create as many projects as you wish. However, you can only work with one project at a time. If you wish to work with another open or created project, select it from the Projects list box.

External Tools Toolbar

From the main toolbar shortcut menu, select the **External Tools** to open this toolbar. External tools configuration from the **Environment Options** dialog box is visible in a compact user-friendly form. Click the arrow button to launch any external tool from the list.



To configure external tools, go to the **Options** main menu, select **Environment**, then select the **External Tools** pane. For more information, see the section “External Tools” on page 156.

Using the Browser

[View Online Demo](#) MagicDraw Basics

The Browser provides a visual representation of the hierarchy of your model elements. The items in this hierarchy are either:

- Compressed - a plus sign next to an icon indicates that the icon is compressed, which means that it contains other model elements. This is the default setting when you start your application. Click the plus sign to expand the icon and view its subordinate items.
- Expanded - a minus sign next to an icon indicates that the icon is fully expanded. Click the minus sign to collapse the item.

If there is no sign next to an icon, it does not contain other model elements.

The Browser is a hierarchical navigation tool that allows you to manage your model data, including packages, components, classes, all UML diagrams, extension mechanisms, and other data. The Browser may be used as an alternative tool to the menus and toolbars that are in MagicDraw. It is easier to work with project diagrams and data elements using the Browser. The Browser performs the following operations:

- Creation and specification of model elements without viewing them.
- Copying, cutting, and pasting of model elements.
- Opening and deleting of model elements.
- Dragging and dropping of model elements to the Diagram pane and inside the Browser.
- Dragging and dropping of data in the Code engineering sets (you may create data in the Data branch, drag it to the Code Engineering sets, and then the round trip object is created automatically).
- Hierarchical viewing of all model elements.
- Trace viewing for the selected model element.
- Symbol creation for the selected model element in the current diagram.
- Managing diagrams.
- Managing extension mechanisms such as constraints, stereotypes, and tagged values.
- Java reversing of a class directly from the classpath.
- Adjusting the code engineering sets.
- Code generation for particular sets.
- Filtering of the visible items, by any model type,(e.g. class, package, operation, component, state and others - for both views and dates), when the Filter from the Browser shortcut menu is selected.
- Sorting of the visible items for the selected model element.
- Sorting of all model elements.
- Displaying search results.

The Browser window parts

The Browser window is divided into two parts:

- **Containment tree/Diagrams tree/Inheritance tree/Model Extensions tree/Search Results.**
The Containment tree tab groups data in the logical sets.
The Diagrams tab groups diagrams that are represented on the diagram pane according to the diagrams type or shows them as a list.

The Inherence tree tab represents the class hierarchy of the project.

The Model Extensions tree tab represents all predefined and created constraints and stereotypes.

The Search results tab displays search results.

- **Documentation/Zoom Control** part. The Documentation tab shows documentation associated with the selected item. The **Zoom Control** tab is responsible for zooming the current diagram.

3 USING MAGICDRAW

Using the Browser

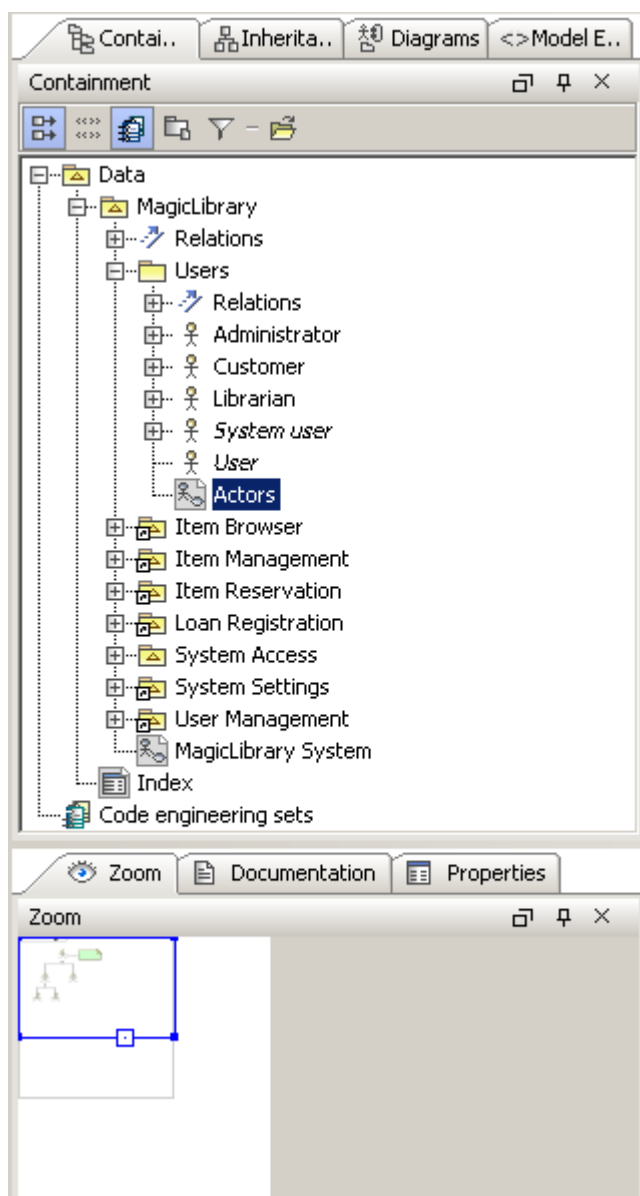


Figure 26 -- Browser

When at least one Project is open, the Browser is placed to the left side of the main window. Beginning with MagicDraw version 7.5, it is possible to move the Browser to any place on the the MagicDraw application. Also, all tabs can be viewed separately and you may set up the Browser according to your needs by hiding the desired tabs.

To change the size of either part of the Browser

Drag the bar that separates the two parts.

To change the Browser position from the **Options** menu

1. Select **Environment** from the **Options** menu. The **Environment Options** dialog box opens.
2. In the Browser tab, change the **Browser Position** property to **Right** or **Left**.

To close or reopen the desired tab of the Browser

From the **Window** menu, choose the tab you want to close/open.

To reset all Browser tabs to the default position

From the **Window** menu, choose **Reset Windows Configuration** command.

To sort items in browser alphabetically

1. Select **Environment** from the **Options** menu. The **Environment Options** dialog box opens.
2. In the **Browser** tab, set the **Sort Always** check box to "true" (default "true"). The same option is available from the **Browser** shortcut menu.

Containment tree

The Containment tree displays model data, grouping it in logical sets.

To open the Containment tree

- Click the **Containment Tree** tab at the top of the Browser.
- If the Containment tree is hidden, from the **Window** menu, select **Containment Tree**.

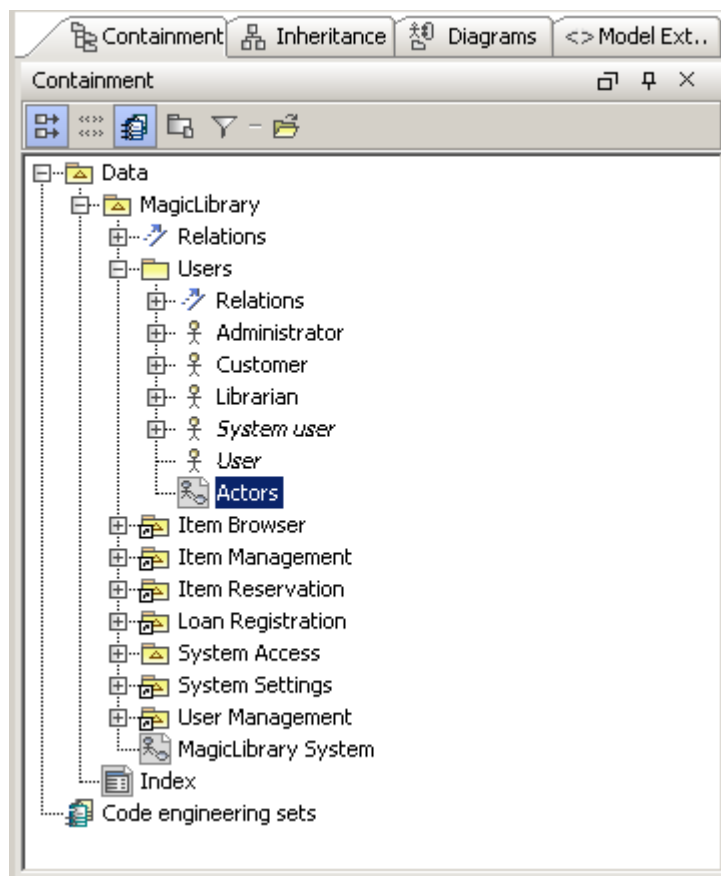




Figure 27 -- Containment tree


To show full information of operations, attributes, and relationships in the Containment Tree

- In the **Containment Tree** tab, click the **Show Full Types in Browser** button  .
 1. Choose **Environment** command from the **Options** menu. The **Environment Options** dialog box appears.
 2. In the **Browser** tab, set the **Show Full Types in Browser** check box to "true" (default "false").


To show stereotypes in the Containment tree

- In the **Containment Tree** tab, click the **Show Stereotypes in Browser** button  .
- 1. Select **Environment** from the **Options** menu. The **Environment Options** dialog box opens.
- 2. In the **Browser** tab, set the **Show Stereotypes in Browser** check box to “true” (default “false”).

To show/hide Code Engineering sets branch

- In the **Containment Tree** tab, click the **Show Code Engineering Sets** button  .
- 1. Select **Environment** from the **Options** menu. The **Environment Options** dialog box opens.
- 2. In the **Browser** tab, set the **Show Code Engineering Sets** check box to “true” (default “true”).

To show/hide Modules

- In the **Containment Tree** tab, click the **Show Modules** button  .
- 1. Select **Environment** from the **Options** menu. The **Environment Options** dialog box opens.
- 2. In the **Browser** tab, set the **Show Modules** check box to “true/false” (default “true”).



To filter types of elements to be displayed

To improve accessibility the **Filter** button has been added to the Containment Tree toolbar (Figure 28 on page 112). Previously the **Filter** menu was accessible only from the Containment tree shortcut menu (Figure 29 on page 113).

The **Items Filter** dialog allows you to choose what types of elements to be displayed in the Containment tree (Figure 30 on page 114).

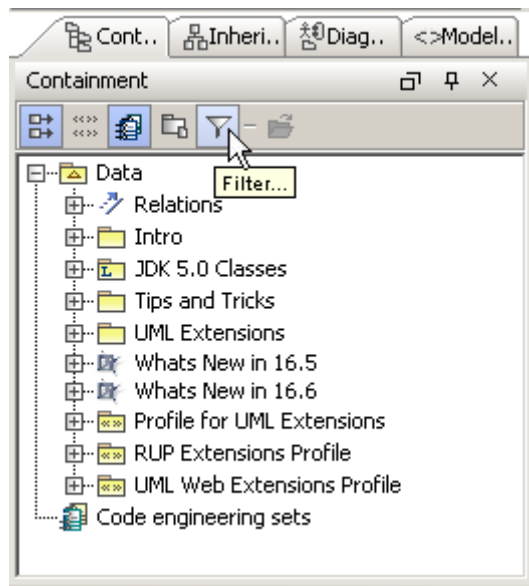


Figure 28 -- The Filter Button in the Containment Tree

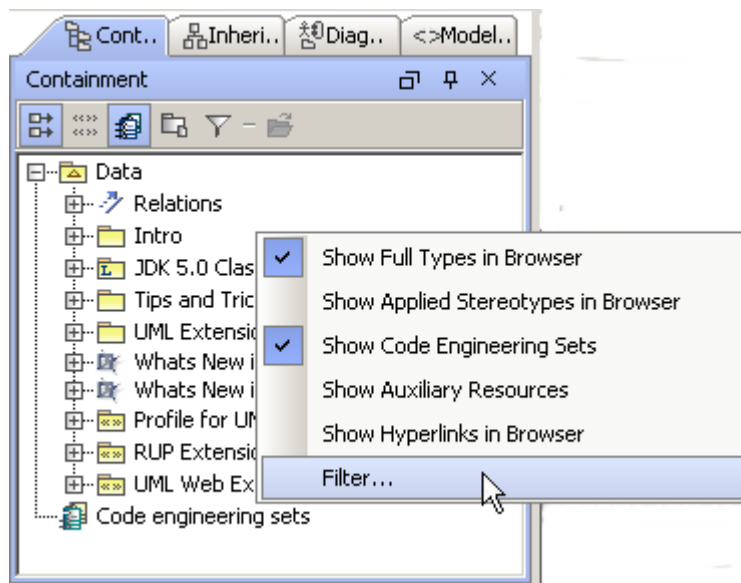


Figure 29 -- The Filter Command in the Browser Shortcut Menu

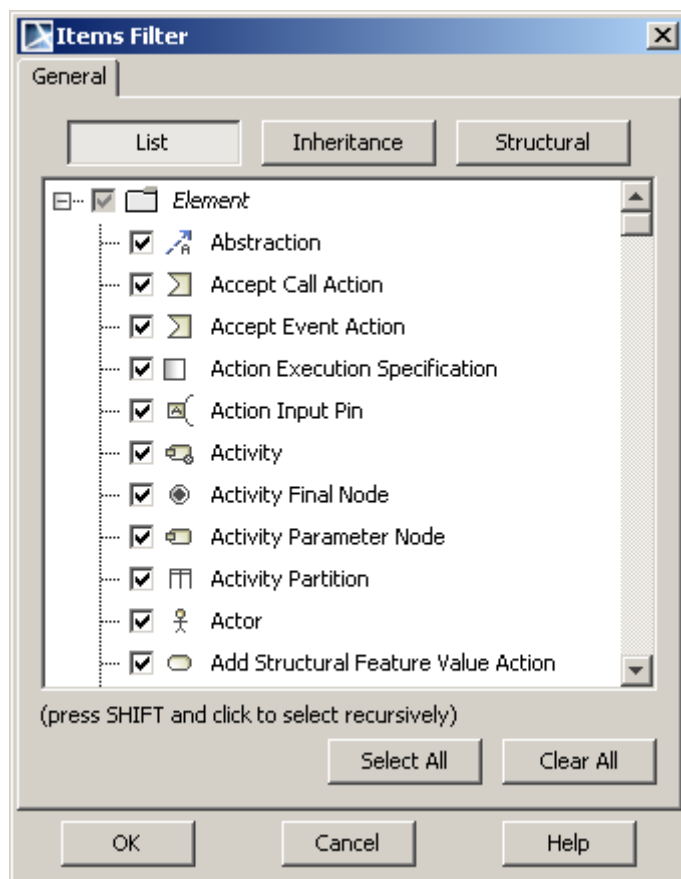


Figure 30 -- The Items Filter dialog box

To open package contents in a new tab

- In the **Containment Tree** tab, click the **Open in New Tab** button . New tab with package name and tree contents will be opened in the Browser.

Data branch

The **Data** branch represents the model and structure of a project. All model elements are stored in packages. This helps you distribute data into logical groups. By default, all new model element data

(inner structure) are stored in the **Data** package. You may create your own packages for storing your model element data.

The **Data** branch also contains the **File View** package, **UML Standard Profile** with stereotypes and data types, and **Relations** package (appears only when at least one path is drawn on the diagram pane).

The **File View** package is where the components are placed during code engineering.

The **UML Standard Profile** contains a list of stereotypes, data types, and elements from the UML 2 metamodel.

To create a new element

1. Select **New Element** from the package in the **Browser** shortcut menu and then select the desired element from the list.
2. Type the name of the element directly in the Browser tree.

For more information about managing model elements from the Browser, see “Working with Model Elements in the Browser Tree” on page 124.

Code engineering sets

The Code engineering sets branch is a gateway between your source code and model data. Using sets, you can perform Java, C++, IDL, DDL, EJB, CIL, and C# round-trip code engineering (code generation + reverse engineering).

To create a new Code engineering set

1. Right-click the **Code Engineering Sets** item and select **New** from the item shortcut menu or open the **Code Engineering Sets** dialog box and click the **New** button. The **New Set** dialog box opens.
2. Type the set name and select the programming language from the drop-down list (by default - Java).
3. Click **OK** to finish the set creation.

To edit the selected set

Select **Edit** from the set shortcut menu. The **Round Trip Set** dialog box opens. Add/remove files or classes from the Code engineering set.

To rename the selected set

Select **Rename** from the set shortcut menu and type the set name.

To change the code generation properties

Select **Properties** from the set shortcut menu. The **CG Properties Editor for Data** dialog box opens.

For detailed description of the CG Properties Editor dialog boxes, see MagicDraw Help.

To delete the selected set

Select **Delete** from the set shortcut menu.

To restore the deleted set

From the **Edit** menu, select **Undo** or press the shortcut keys CTRL+Z.

To generate code from the selected set

1. Select **Generate** from the set shortcut menu. The **Code Generation Options** dialog box opens.
2. Adjust the code generation options.
3. Click **OK**. The **Messages Window** dialog box opens. Information about generated files is shown.

To check syntax

Select **Check Syntax** from the set shortcut menu. If no errors are found, then a message box opens stating there are no syntax errors in the model.

To reverse the selected set

1. Select **Reverse** from the set shortcut menu.
2. The **Reverse Options** dialog box opens. Define options and click **OK**.
For the detailed descriptions about the reverse, see Section Reverse in the Code Engineering User's Guide.

To reverse files that have been changed

1. Select **Refresh** from the set shortcut menu.
2. The **Reverse Options** dialog box opens. Define the options and click **OK**.
For detailed description about the reverse process, see the “Reverse” Section in the Code Engineering User’s Guide.

TIP! All functions listed above can be performed in the **Code Engineering Sets** dialog box.

To select a text editor for source code

1. In the **Options** menu, select **Environment**, then select the **Launchers** group in the dialog box that opens.
2. In the **Default launchers** field, click the “...” button and select the directory where the text editor is located.
3. Click **OK**.

Diagrams tree

The Diagrams tree in the Browser represents the external structure of a diagram.

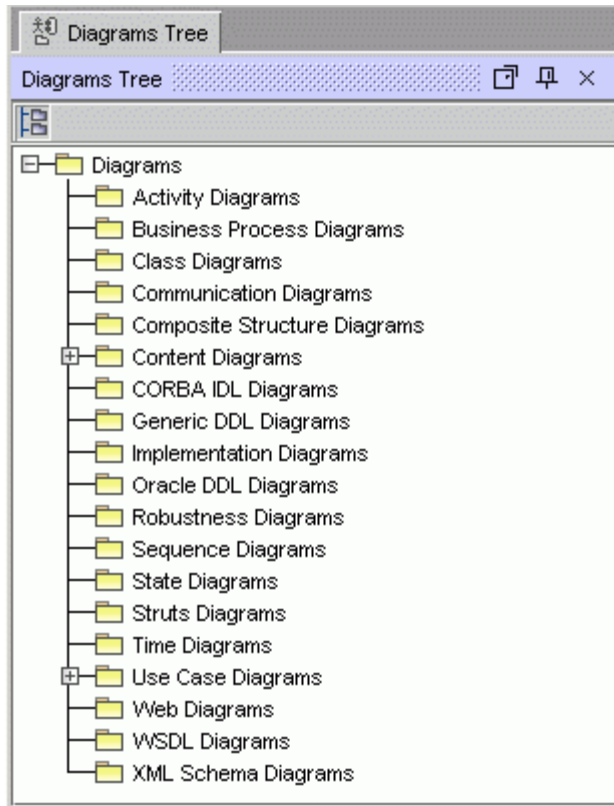



Figure 31 -- Diagrams tree

In the Diagrams tree, with the selected diagrams, you can perform the operations listed below.

To group diagrams according to their diagram type

- Click the **Group by Diagram Type** button  in the Diagrams tree.
- From the shortcut menu, select the **Group by Diagram Type** check box.

To open the selected diagram from the Browser

Select **Open** from the item shortcut menu or double-click the item in the diagram.

To delete the selected diagram

Select **Delete** from the selected diagram shortcut menu.

To rename the selected diagram

Select **Specification** from the diagram shortcut menu. The corresponding **Diagram Specification** dialog box opens. Type the diagram name and click **OK**.

TIP!

In the **Diagram Specification** dialog box you can add documentation to the diagram, view the relationships in which the diagram participates, and define hyperliks, stereotypes, constraints, and tagged values.

To print the selected diagram

- Select **Print** from the diagram item shortcut menu. If the diagram is empty, it will not be printed.

Inheritance tree

The Inheritance tree represents classifiers, packages, data types, and stereotypes hierarchy within your project. Inheritance according to the UML Specification is shown using the generalization relationship.

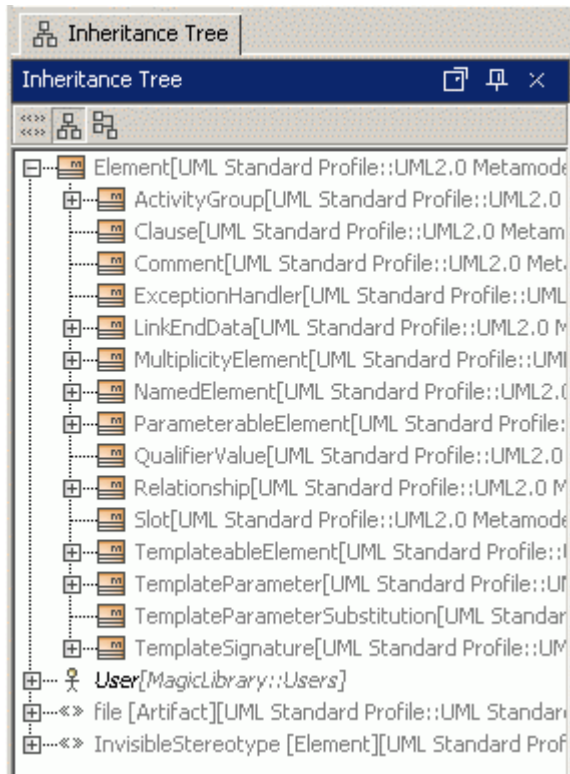



Figure 32 -- Inheritance tree


For more information about managing model elements from the Browser, see “Working with Model Elements in the Browser Tree” on page 124.

To show stereotypes in the Inheritance tree


- In the **Inheritance Tree** tab, click the **Show Stereotypes in Browser** button .
- 1. Select **Environment** command from the **Options** menu. The **Environment Options** dialog box opens.

2. In the **Browser** tab, set the **Show Stereotypes in Browser** check box to “true” (default “false”).

To show classifier hierarchies in the Inheritance tree

- In the **Inheritance Tree** tab, click the **Show only Hierarchies** button . If the classifier has no generalization relationship, it will not be visible on the tree.
1. Select **Environment** command from the **Options** menu. The **Environment Options** dialog box opens.
 2. In the **Browser** tab, set the **Show Only Hierarchies** check box to “true/false” (default “false”).

To invert tree in the Inheritance tree

- In the **Inheritance Tree** tab, click the **Invert Tree** button . The current view in the Inheritance tab shows classifiers, more specific classifiers are shown as their children. After inverting a tree, the classifiers tree view will be change to show the child as a root classifier.
1. SelectChoose **Environment** command from the **Options** menu. The **Environment Options** dialog box opens.
 2. In the **Browser** tab, set the **Invert Tree** check box to “true/false” (default “false”).

Model Extensions Tree

The Model Extensions Tree contains all Stereotypes that are predefined and created manually in the project.

In this tree you can create, review, copy/paste, and delete extension mechanisms.

It is mainly used for the work of a team using Teamwork server for locking for edit/unlocking extension mechanisms.

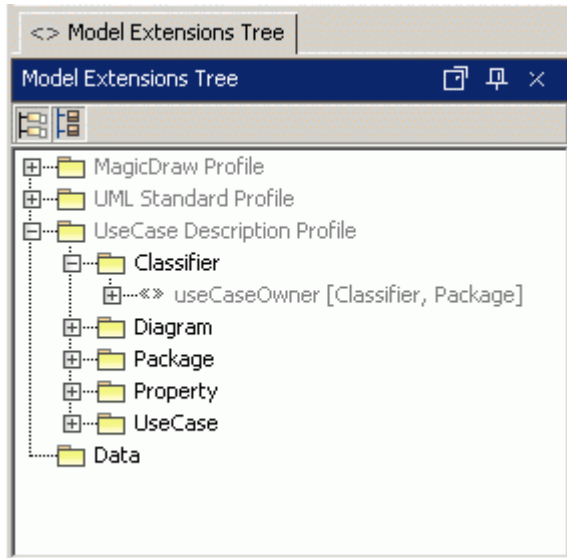




Figure 33 -- Model Extensions Tree

To group extensions by profiles

- In the Model Extensions Tree, click the **Group by Profiles** button .
- From the Model Extensions Tree shortcut menu, select the **Group by Profiles** check box.

To group extensions by metaclasses

- In the Model Extensions Tree, click the **Group by Metaclasses** button .
- From the Model Extensions Tree shortcut menu, select the **Group by Metaclasses** check box.

Search Results Tree

The Search Results tree shows results of a search, which may be performed through the [Find dialog box](#).

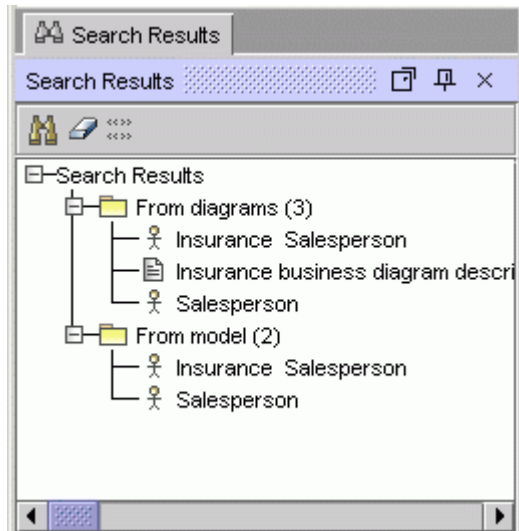


Figure 34 -- Search Results tab

The results in the tab are displayed in two packages:


- From Diagrams - elements are shown that are displayed on the diagram(s).
- From Model - elements are shown that are created in the model data.

For information about how to perform a search, see “Searching” on page 189.

To open the **Find** dialog box from the **Search Results Tree**

Click the **Find** button , or press CTRL+F keys.

To clear the results of the previous search

Click the **Clear Results** button  or select the **Clear Results** button from the **Search Results** tab.

Working with Model Elements in the Browser Tree

To create a model element or a diagram in the Browser

1. Right-click the package in the Browser.
2. From the **New** command, select the model element or diagram you wish to create.

To copy/cut and paste the selected model element in the Browser tree

1. Select **Copy** or **Cut** from the item shortcut menu.
2. Select the item where you wish to put the model element.
3. Select **Paste** from the item shortcut menu.

To copy/cut and paste the selected model element among different projects

1. Select **Copy** or **Cut** from the item shortcut menu.
2. Change the project name in the Projects list on main window toolbar.
3. Select where you wish to put the element.
4. Select **Paste** from the item shortcut menu.

To delete a model element from the Browser tree

Select **Delete** from the item shortcut menu.

To drag-and-drop the selected item in the Browser tree

1. Make sure that the place you wish to drag the item is visible.
2. Drag the selected item to the destination and drop it.

To draw a symbol on the diagram

Select **Create Symbol** from the items in the Browser shortcut menu.

NOTE You may draw a symbol by dragging and dropping an item to the Diagram pane.

To show/hide the model elements in the Browser tree

1. Right-click in the Browser. From the shortcut menu, select **Filter**.

- Or press the **Filter** button in the Containment tree.
- 2. The **Items Filter** dialog box opens. Clear the check boxes of those model elements that you would not like to appear in the Browser tree.

For more information about how to filter items, see “To filter types of elements to be displayed” on page 111.

Multiple selection

A group of model elements can be selected within the Browser tree and you can edit all the selected model elements at the same time.

To make multiple selections

- Hold down the SHIFT key and click the last element you wish to include in the multiple selection.
- For more precise selection, hold down the CTRL key and click (while holding the key down) with the mouse on the elements you wish to select.

To select all model elements or all browser tree items

Click the mouse pointer in the area you want to select all elements and press the shortcut keys CTRL+A.

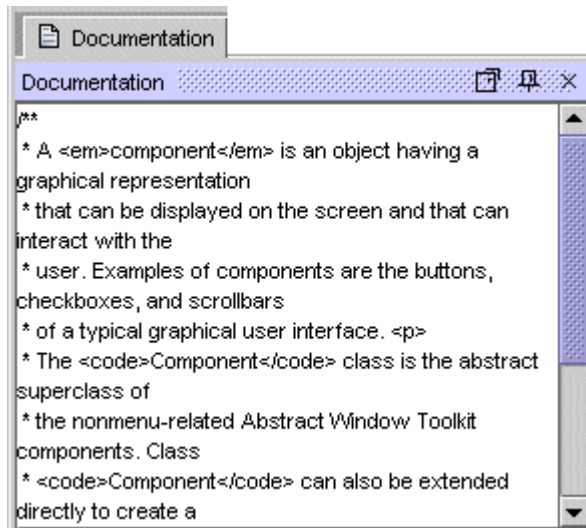
NOTE

All selected model elements can be moved or deleted as a single selected element. While moving the group of elements, a border appears denoting the area you have selected. Be careful when deleting multiple elements because no confirmation dialog box will appear.

Documentation/Zoom Control/Properties

Documentation tab

The Documentation tab shows information associated with the selected model element in the Browser tree or diagram.



To open the Documentation tab

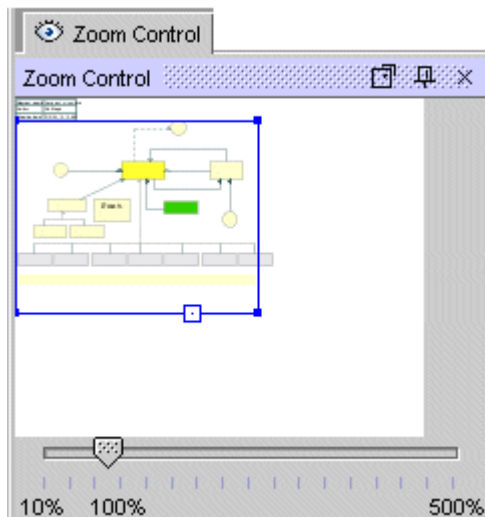
Click the **Documentation** tab in the lower side of the Browser window or from the **Window** menu, select **Documentation**. If there is no documentation for the selected element, no documentation message is displayed in the section.

To write documentation for the selected model element

1. Click the Documentation tab.
2. Type information.

Zoom Control tab

In this tab, you can preview the desired diagram by selecting it in the Browser on the Content diagram. Also you use this tab for zooming the diagram.



To zoom in for a close-up view of your diagram, or to zoom out and see more of the diagram at a reduced size

1. Click the **Zoom** tab in the Browser window.
2. Drag the rectangle.

NOTE

By default the zoom slider is not visible. To display the zoom slider in the **Zoom** tab, in the **Environment Options** dialog box, **Browser** tab, select the **Show Diagram Zoom Slider** check box.

Quickly access any part of the diagram using the **Zoom** tab

Drag the blue square to the desired part of the diagram. The desired diagram part will be displayed in the diagram window.

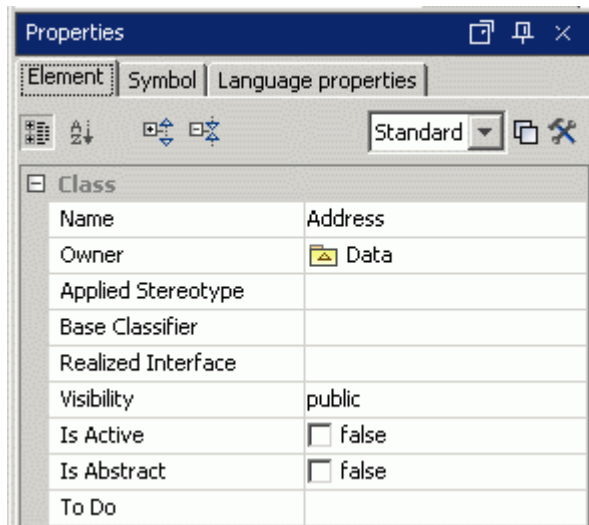
To fit the diagram to the window using the **Zoom** tab

Click the **Fit in Window** button  on the square in the **Zoom** tab.

Properties tab

Since MagicDraw version 8.0, you can edit model elements and diagram properties such as data, symbol properties, or language properties directly from the Browser, **Properties** tab.

The Properties tab has two modes - Standard and Expert. Choose the mode that best suits your needs.



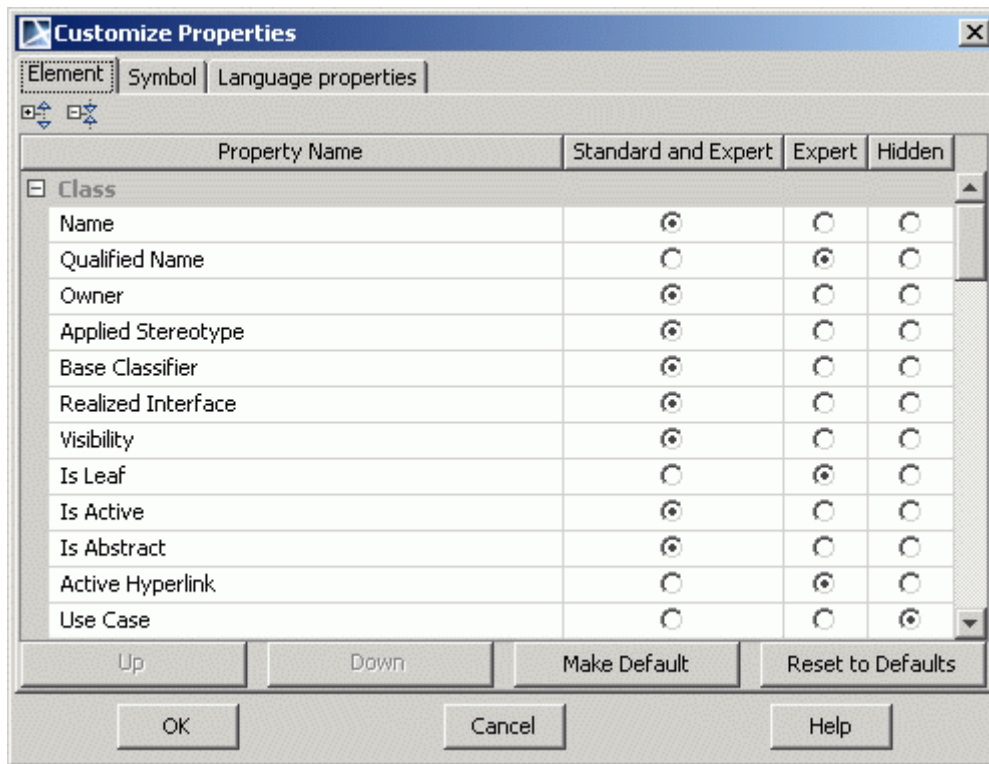


Figure 35 -- Customize Properties dialog box

Setting Environment Options

You can customize MagicDraw according to your preferences: Change the application settings in the **Environment Options** dialog box.

To open the **Environment Options** dialog box

Select **Environment** from the **Options** menu.

Buttons available in the **Environment Options** dialog box

Reset to Defaults	Resets all properties to the default settings.
OK	Saves changes and exits the dialog box.
Cancel	Exits the dialog box without saving changes.
Help	Displays MagicDraw Help.

General pane

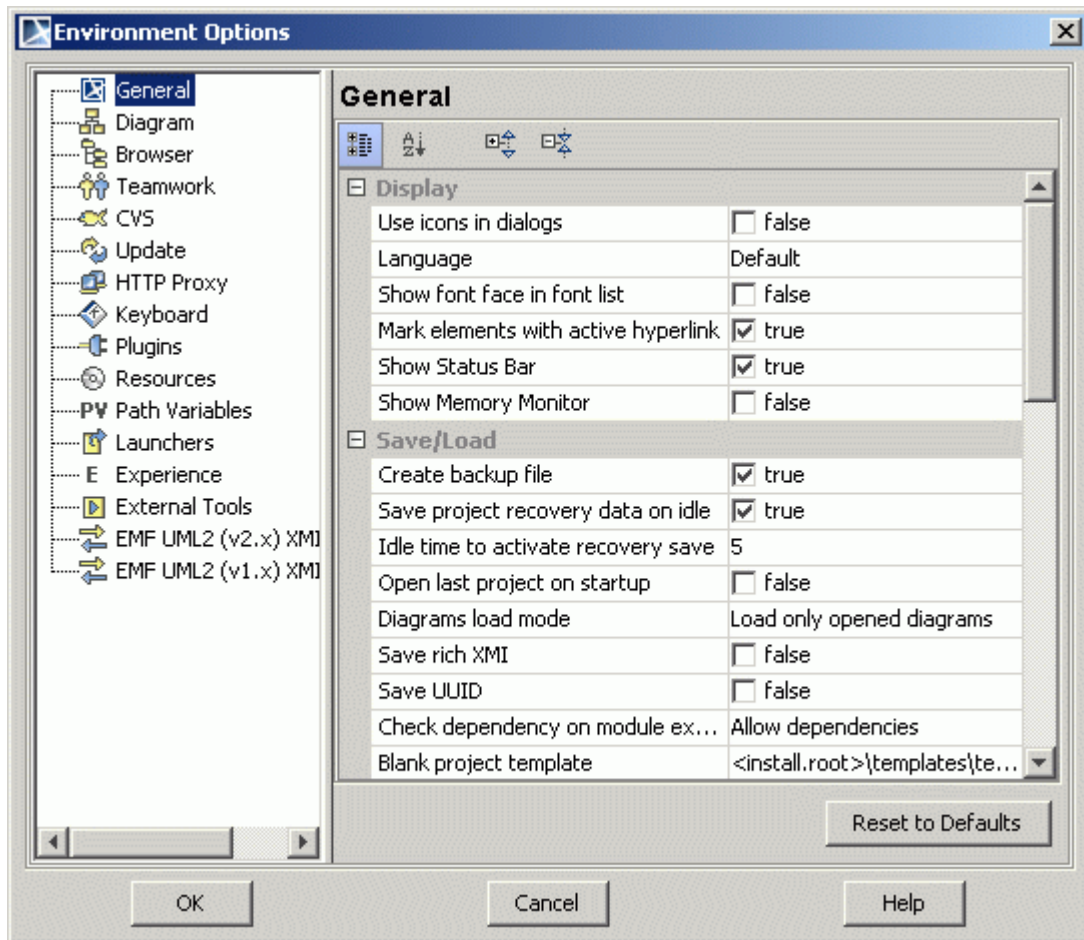


Figure 36 -- Environment options dialog box. General pane

Change the appropriate fields according to your preferences. If the check box is selected, the description False changes to True.

Property name	Function
Use icons in dialogs	Icons are shown on the button of dialog boxes.

Property name	Function
Language	<p>Select the language for the MagicDraw GUI:</p> <ul style="list-style-type: none"> • Default – according to your application. It is set by default. • English • French • German • Italian • Japanese • Korean • Portuguese Brazil • Spanish • Thai <p>NOTE Restart MagicDraw to apply changes to the language property.</p>
Show font face in font list	Font names that are listed in the Font drop-down list box on the main toolbar will be displayed in the original font.
Mark elements with active hyperlink	Shows active hyperlink arrow on the element symbol.
Show status bar	Shows status bar line at the bottom of the MagicDraw window.
Show memory monitor	Shows the project memory usage line at the bottom of the MagicDraw window.
Create backup file	Backup files of your projects will be created. Backups are saved with the name pattern old_name.xml.bak or old_name.xml.zip.bak.
Save project recovery data on idle	Saves AutoRecovery file of the open project(s) when a system is not in use.
Idle time to activate recovery save	Enter the system idle time (in minutes). This is the amount of time until the file recovery save is activated.
Open last project on startup	The last project of your previous session will be opened next time you start MagicDraw.
Diagrams load mode	<p>Select how to load diagrams after opening a MagicDraw project.</p> <ul style="list-style-type: none"> • Load all Diagrams – opens all diagrams that are in the project. • Load Only Open Diagrams (<i>default</i>) – opens only diagrams that were not closed in the previous use of the project. • Do not Load Diagrams – no diagrams will be opened when opening the project.
Save rich XMI	Saves maximum additional information to an xmi file, which is not required for MagicDraw load but may be needed when using other tools.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Save UUID	
Check dependency on module export/share	Possible choices: <ul style="list-style-type: none">• Do not check• Allow dependencies (default)• Do not allow dependencies
Blank project template	Default location and template file name: <install.root>\templates\template.xml
Save settings on exit	The altered settings within the current MagicDraw session will be stored and applied for all future sessions.
Show tip of the day	Shows the Tip of the Day dialog box every time you start MagicDraw.
Recent files list size	Type the number of projects to appear in the File menu. Click the file name to open it. The maximum file list size is 10. The file name list shows files that were used in MagicDraw.
Recent windows list size	Type the number of recently used windows that are listed in the Window menu. Click the window name to make it active. The maximum windows list size is 10.
Undo list size	Type the number of actions that will be available for undo/redo operations. They will be listed in undo/redo drop down lists on the main toolbar. The maximum undo/redo list size is 500.
Save diagram background in image	Saves the diagram with background. By default, the diagram background is white after saving as image.
JPEG Compression Quality	This is a JPEG format option.
Use SVG <text> tag for text output	This is an SVG format option.
Image resolution (DPI)	This property is DPI property value with numeric value range from 1 to 4800. Default value is 72.
Exported image size [%]	Define image size in percent. Default value is 100%. For example; if 200% is defined, then the view is enlarged (zoomed) before generating an image. Raster image will not loose quality as additional pixels are introduced.
TIFF Color Space	This is a TIFF format option.
TIFF Compression	This is a TIFF format option.
Help font	Choose the font that will be used in the Help window.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Help server port	The default Help server port is 1111. Change it if some other server is using this port.
Use model enrichment wizard	<p>The Use Case Model Wizard dialog and the Software Design Model Wizard dialog are model enrichment wizards.</p> <p>To invoke the Use Case Model Wizard dialog from the Tools main menu, choose the Use Model Enrichment Wizard command.</p> <p>To invoke the Software Design Model Wizard dialog from the Tools main menu, choose the Software Design Model Enrichment command.</p> <p>For more information about reports generation see "<i>MagicDraw ReportWizard UserGuide.pdf</i>" which is in <MagicDraw installation directory>\Manuals folder.</p>
Show model enrichment suggestion dialog	For more information about reports generation see " <i>MagicDraw ReportWizard UserGuide.pdf</i> " which is in <MagicDraw installation directory>\Manuals folder.
Specification opening mode	<p>Possible choice:</p> <ul style="list-style-type: none">• In the same window• In a new window
Recent specifications list size	Specify the list size for specifications. Possible range 1 to 20.
Specifications history size	Specify the history size for open element specifications. Possible range 1 to 20.
Opaque Expression default language	<p>Possible choices:</p> <ul style="list-style-type: none">• English• OCL
Opaque Expression displaying lines	<p>Possible choices:</p> <ul style="list-style-type: none">• 1• 2• 3• 4• 5• All Lines
Hide toolbars in full screen mode	The main toolbar is hidden when the diagram full screen mode is turned on. To display the main toolbar, clear the Hide toolbars in full screen mode check box in the Environment Options dialog box, General pane, General group.

Diagram pane

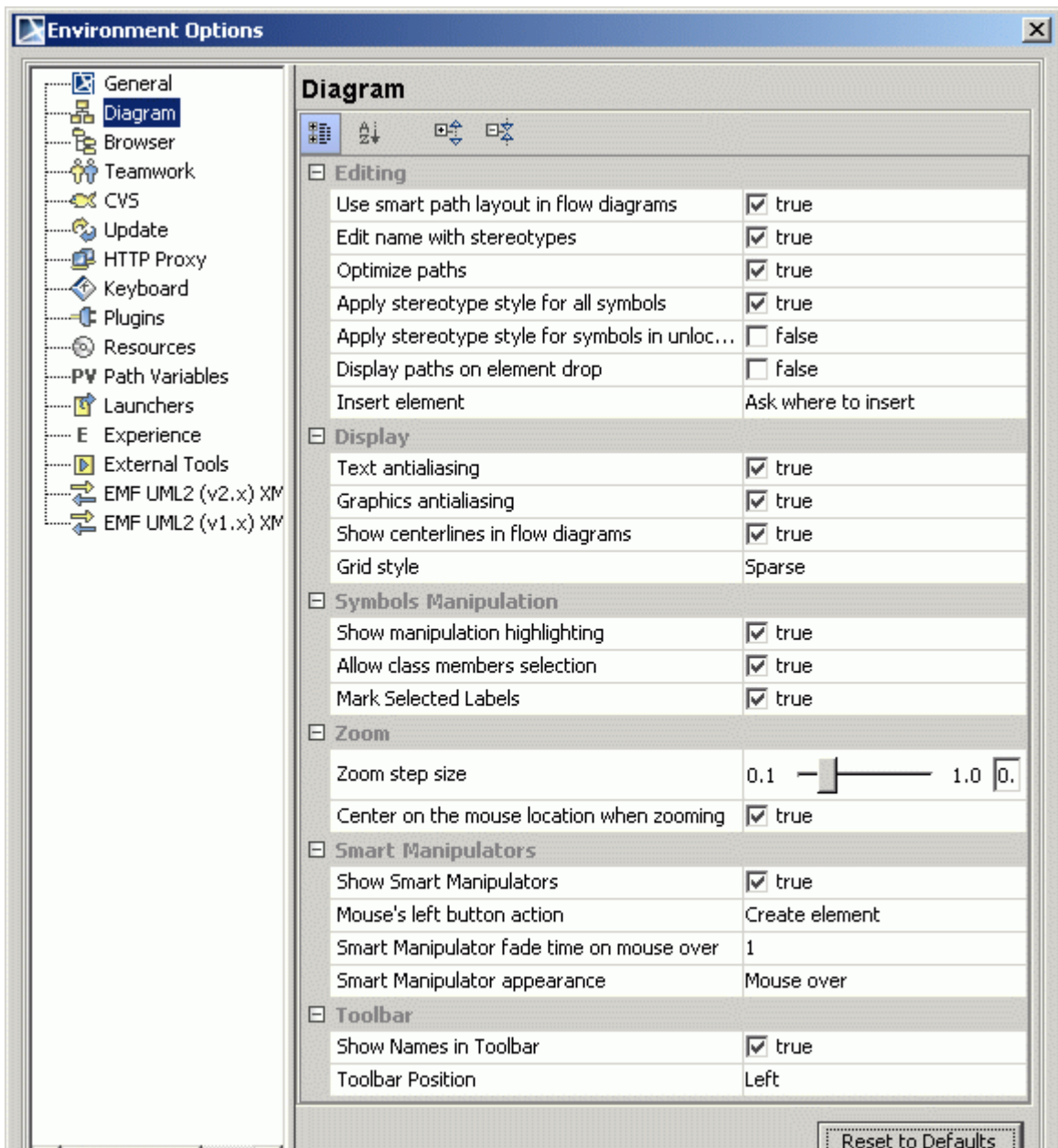


Figure 37 -- Environment Options dialog box. Diagram pane

Property name	Function
Text antialiasing	Smooths the jagged edges of text.
Graphics antialiasing	Smooths the jagged edges of graphics.
Grid style	Select one of the following styles of the grid: Dense or Sparse .
Validate Shape Ownership	Validates if the symbol owner on the diagram matches the actual element owner in the model. Using the symbol ownership validation feature, shape ownership problems will be highlighted on the diagram pane and you will be able to easily see and resolve them. For more information see “Resource Manager” on page 497.
Shape Ownership Validation Period (seconds)	Type the period in seconds to check diagram for symbol ownership problems.
Edit name with stereotypes	Allows the management of stereotypes by editing the stereotype name directly on the Diagram pane.
Show manipulation highlighting	Enable error highlighting of the modeling process. This helps you to see the errors in the model at drawing time. The valid and invalid actions will be highlighted in different colors (defaults are red for invalid actions, blue for valid actions).
Allow class members selection	Operations and attributes can be selected, then dragged & dropped to another class.
Optimize paths	When a path is drawn in various directions, this option removes loops from the path.
Apply stereotype style for all symbols	Applies the created stereotype style to a symbol when adding the stereotype to an element.
Apply stereotype style for symbols in unlocked diagrams	Applies the stereotype style in teamwork for unlocked diagrams.
Mark selected labels	Mark path labels after a path is selected.
Display paths on element drop	When dragging and dropping an element from the Browser or executing a Create Symbol action, the path to the existing symbols is created.

Property name	Function
Insert Element	<p>Splits the path into two paths by drawing a symbol on it and inserts this symbol according to the choice. Possible choices:</p> <ul style="list-style-type: none"> • Before path • After path • Do not insert • Ask where to insert <p>NOTE: This is valid only in state and activity diagrams.</p>
Set preferred size after hiding compartment	<p>Specifies if shape size shall be set to preferred after one of shape's compartments is suppressed in the diagram. Possible choices:</p> <ul style="list-style-type: none"> • Never; • Always; • If no connected paths (default).
Auto completion includes metaclasses	<p>If selected, the list of available elements to choose, element types, or stereotypes includes metaclasses (in MagicDraw metaclasses are placed in the <i>UML Standard Profile</i>. Default value is <i>false</i>.</p>
Auto completion includes elements from profiles and modules	<p>When selected, the list of available elements to choose, element types, or stereotypes includes elements, which are placed in profiles and modules. Default value is <i>true</i>.</p> <p>NOTE: This option toggles all profiles except <i>UML Standard Profile</i>.</p>
Zoom step size	<p>Type in the step value of zooming your diagram views. Step value should be in the range from 0.1 to 1.0.</p>
Center on the mouse location when zooming	<p>Centers zooming according to mouse arrow.</p>
Show Smart Manipulations	<p>Set smart manipulator visibility. For a detailed description about the smart manipulation feature, see "Smart Manipulation" on page 269</p>
Mouse's Left Button Action	<p>Possible choices:</p> <p>Create element - target element is created during the creation of a path after a mouse click on the diagram pane.</p> <p>Make a break point - set a break point for the path after a mouse click.</p>
Smart Manipulator fade time on mouse over	<p>Set the smart manipulators fade time in seconds.</p>

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Smart Manipulator appearance	Possible choices: <ul style="list-style-type: none">• Mouse Over• Shift
Show Names in Toolbar	Shows element names in toolbars near icon.
Toolbar Position	Change toolbar appearance near diagram pane. Possible choices: <ul style="list-style-type: none">• Right• Left

Browser pane

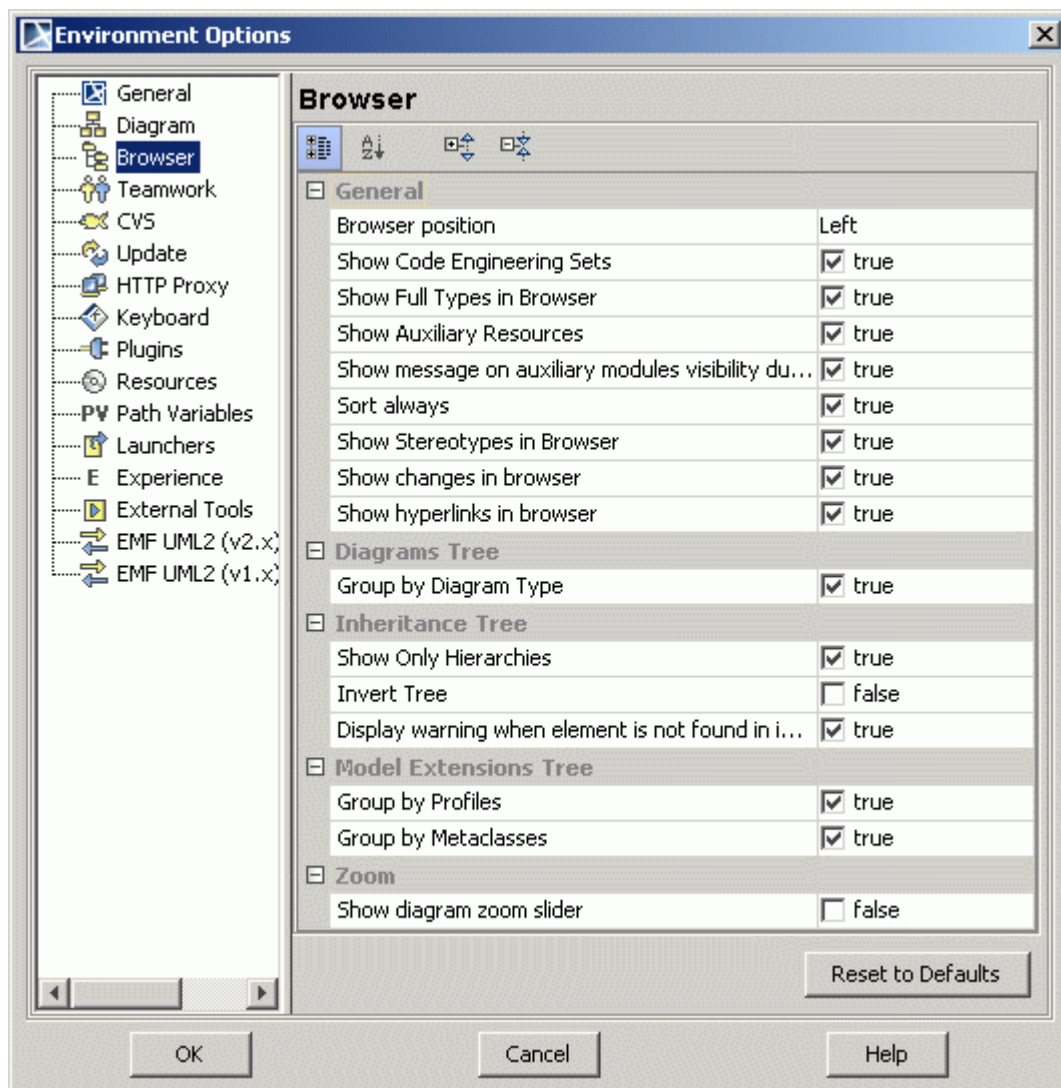


Figure 38 -- Environment Options dialog box. Browser pane

Property name	Function
Browser position	In the drop-down list, select Left if you wish the browser to appear on the left side of the workspace. Select Right if you wish the browser to appear on the right side of the workspace.
Show code engineering sets	Shows code engineering set items in the browser.
Show full types in browser	Shows the full attributes, operations, and relationships data.
Show Auxiliary Resources	Shows/hides the profiles, modules with applied << <i>auxiliaryResources</i> >> stereotype in the Browser.
Show message on auxiliary modules visibility during use	If Show Auxiliary Resources option is false and profiles/modules are not displayed in the Browser, while exporting module, a message will appear.
Sort always	Always sort browser items alphabetically.
Show stereotypes in browser	Shows stereotypes that are assigned to the model elements.
Show changes in browser	Highlight edited or added model elements and diagrams in the Browser.
Show hyperlinks in browser	Shows hyperlinks that are assigned to the model elements.
Group by diagram type	Groups diagrams according to diagram type in the Diagrams tree.
Show Only Hierarchies	When selected, classifiers without hierarchies are not displayed in Inheritance tree. In this case, Inheritance tree displays only classifiers connected with the generalization relationship.
Invert Tree	When selected, hierarchies are inverted so that at the top of the hierarchy, the specific element is displayed.
Display warning when element is not found in inheritance tree	Displays warning when element was not found in the Inheritance tree.
Group by Profiles	Stereotypes, tagged values, and constraints are grouped by profiles in the Model Extensions Tree.
Group by Metaclasses	Groups stereotypes by Metaclass in the Model Extensions Tree.
Show diagram zoom slider	Shows zoom slider in the Zoom Control tab.

Teamwork pane

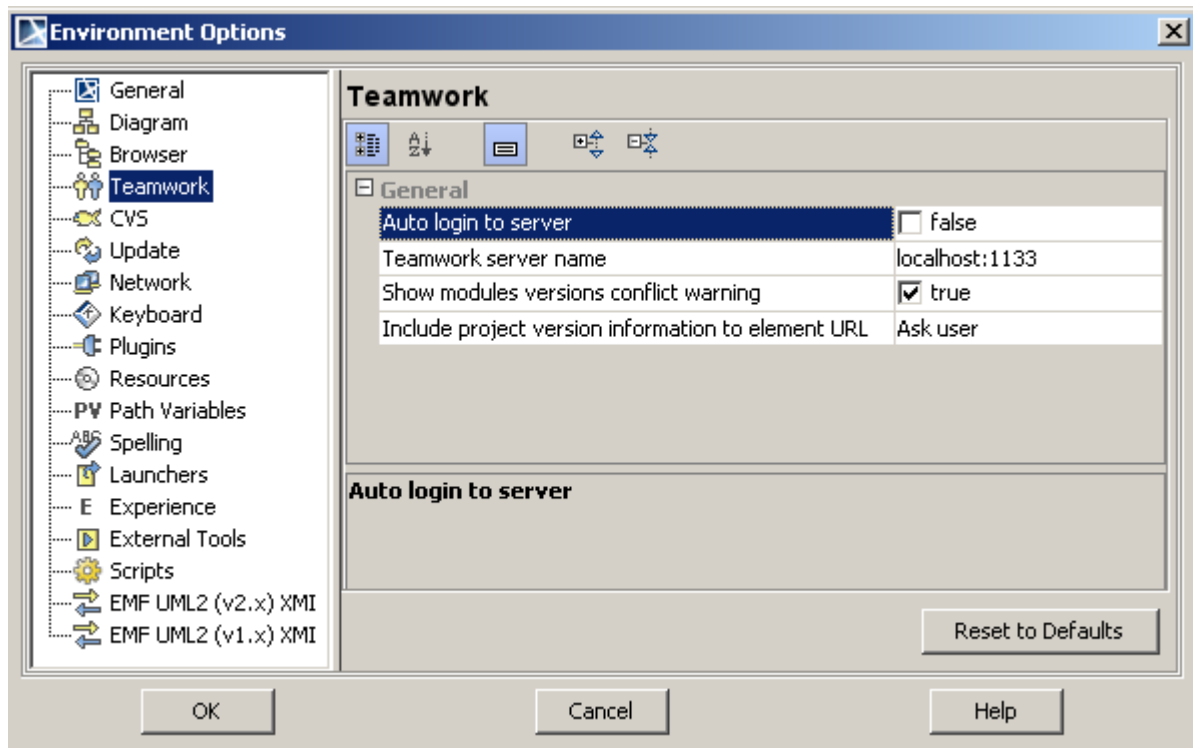


Figure 39 -- Environment Options dialog box. Teamwork pane

Property name	Function
Auto login to server	The user automatically logs on to the Teamwork Server when MagicDraw starts.
Teamwork server name	Shows the name of your teamwork server.
Show modules versions conflict warning	The warning appears when two modules (or module and a project) use the same submodules.

Property name	Function
Include project version information to element URL	You can copy a project element URL to a clipboard and share it with other as a quick reference to model elements. Select option to include project version information to element URL. For more information about Copying/Opening Element URLs, see “Copying/Opening Element URLs” on page 385.

Floating pane

This pane is available only if you use MagicDraw Floating License.

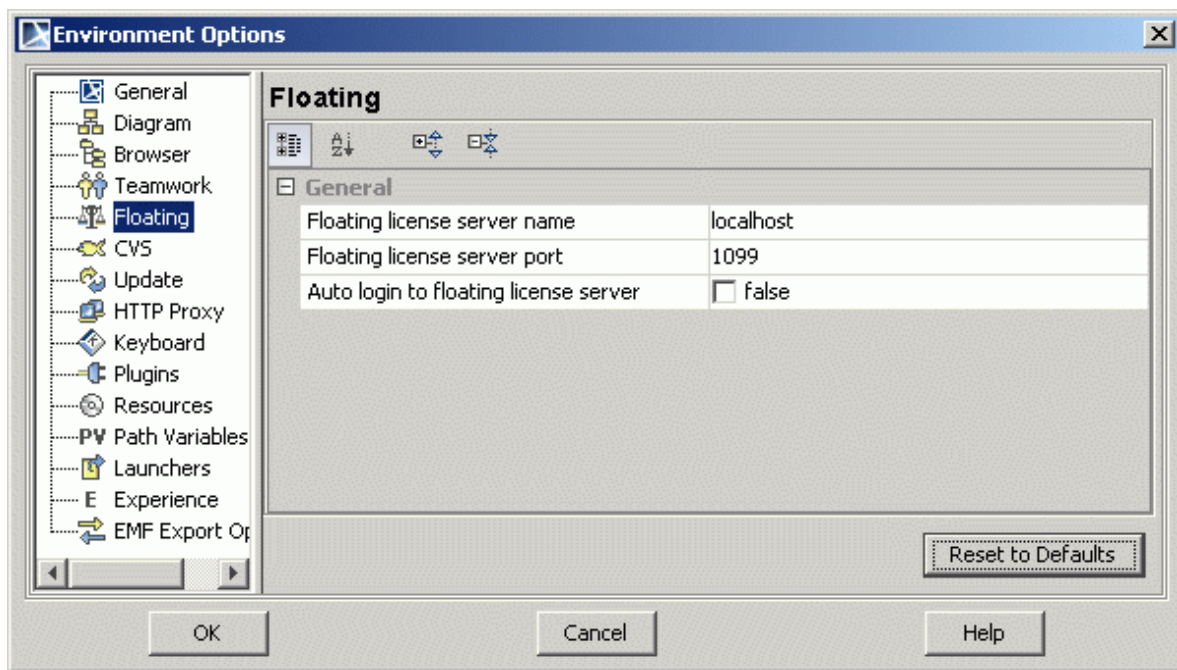


Figure 40 -- Environment Options dialog box. Floating pane

Property name	Function
Floating license server name	Specify the name of the Floating License Server.
Floating license server port	Enter the port number of the Floating License Server. The server may be started on many different ports.

Property name	Function
Auto login to floating license server	If selected, automatically logs in to the Floating License Server when MagicDraw starts.

CVS pane

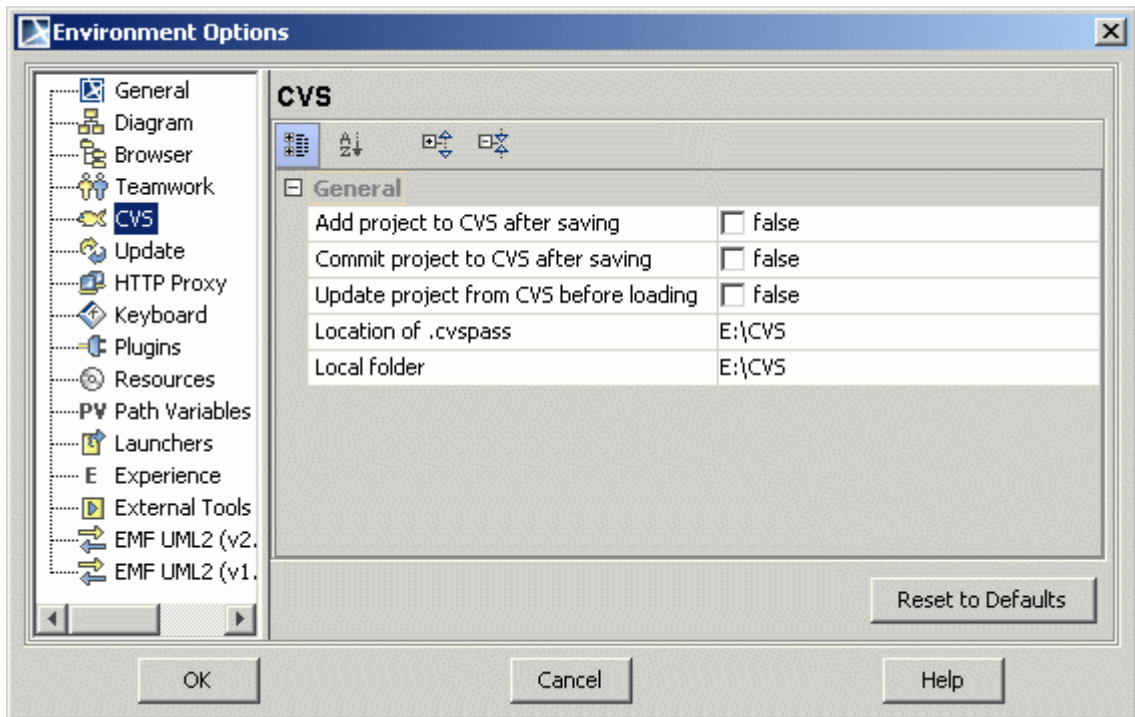


Figure 41 -- Environment Options dialog box. CVS pane

Property name	Function
Add project to CVS after saving	When a new project is saved, it will be added to the CVS checked out directory. The Add Project to CVS dialog box appears.
Commit project to CVS after saving	Commits project to CVS after saving it. The Commit Project to CVS dialog box opens.
Update project from CVS before loading	Updates the project from CVS before loading. The Update CVS Project dialog box opens.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Location of .cvspass	The path where the .cvspass is located. You may type it or choose the path from the Open dialog box.
Local folder	The path where the module will be saved on checkout action. You may type it or choose the path from the Open dialog box.

Update pane

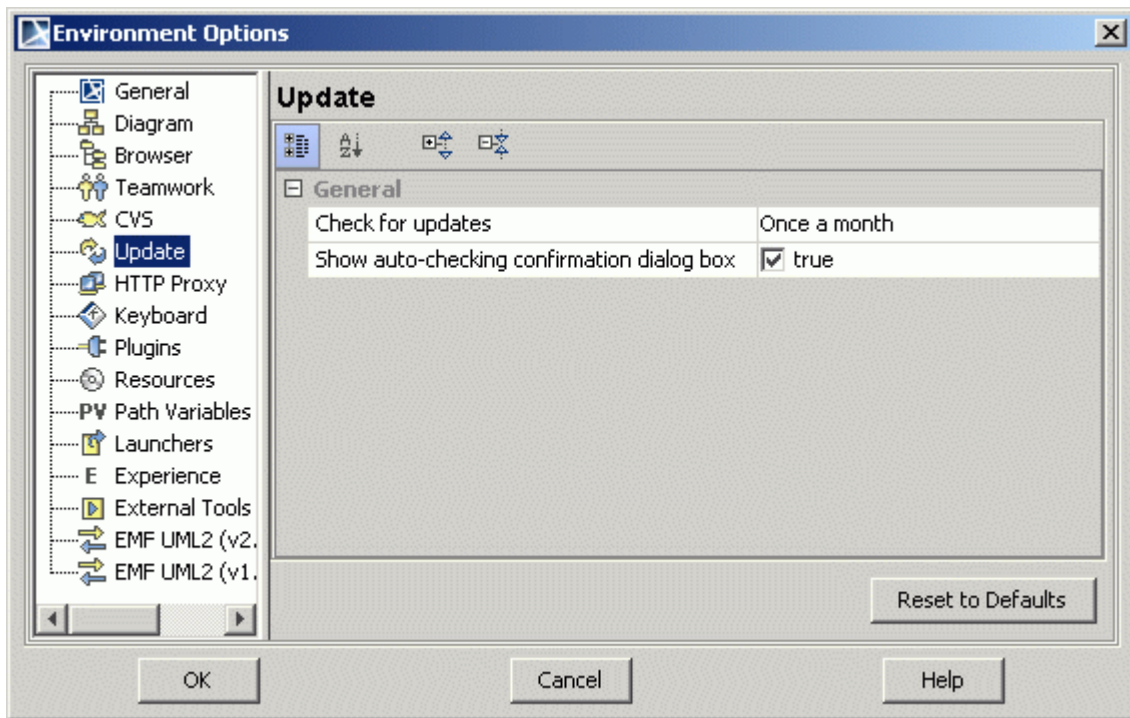


Figure 42 -- Environment Options dialog box. Update pane

Property name	Function
Check for Updates	<p>Select the period for checking MagicDraw for updates.</p> <ul style="list-style-type: none"> • Manually • On startup • Once a day • Once a week • Once a month (recommended)
Show auto-checking confirmation dialog box	<p>If selected, shows the confirmation dialog each time before auto-checking.</p>

Network pane

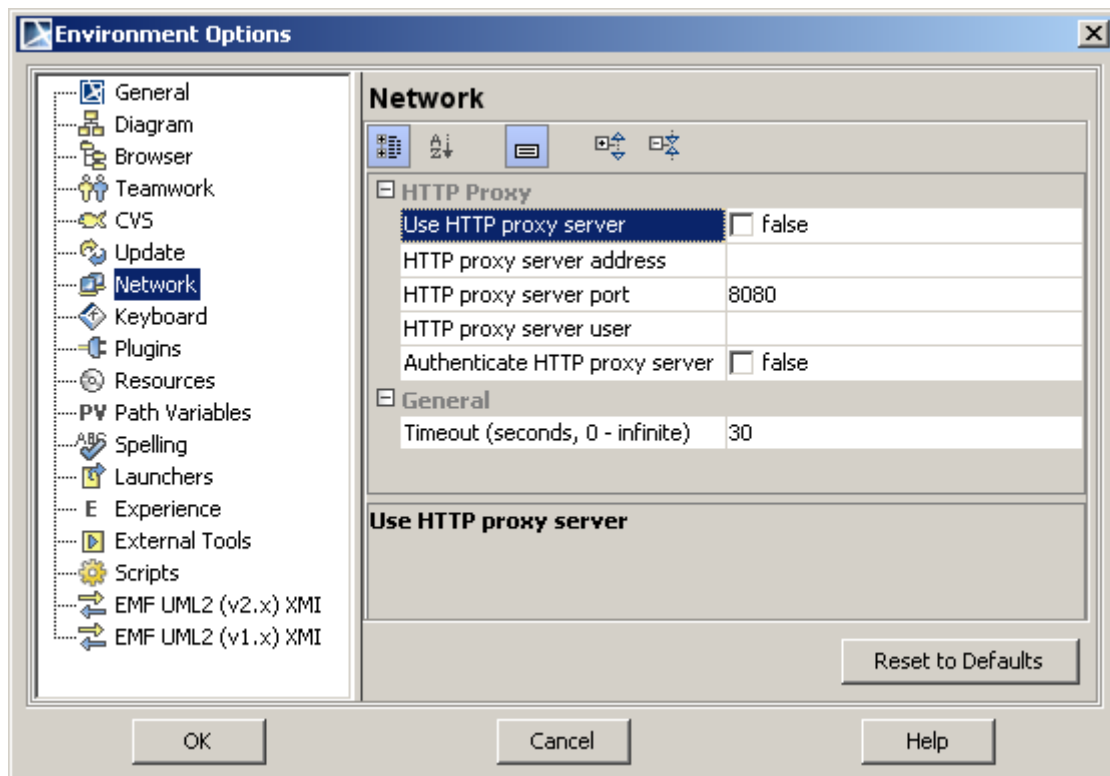


Figure 43 -- Environment Options dialog box. Network pane

Property name	Function
Use HTTP proxy server	If selected, uses the HTTP Proxy Server when MagicDraw is checking for updates.
HTTP proxy server address	Type the HTTP Proxy Server name or IP address.
HTTP proxy server port	Type the port of the HTTP Proxy Server. The default number is 8080.
HTTP proxy server user	Type the name of the HTTP Proxy Server user.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Authenticate HTTP proxy server	Authenticates the HTTP Proxy Server.
Timeout (seconds, 0 - infinitive)	Seconds to wait before cancelling a connection attempt.

Keyboard pane

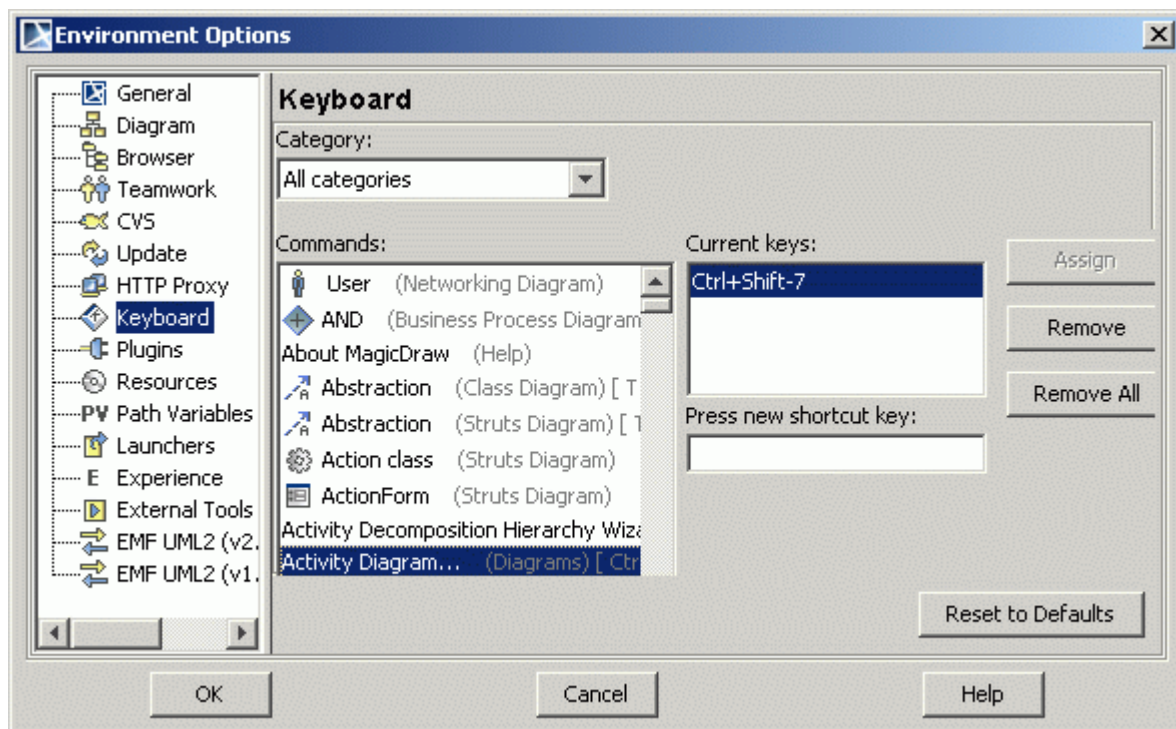


Figure 44 -- Environment Options dialog box. Keyboard pane

Property name	Function
Category	The list of all available categories of the keys (File, Edit, Tools, etc.). The default item in this choice is always All Categories . Selecting the category opens its content in the Commands list.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Commands	The list of commands from the selected category. If All Categories is selected in the Categories list, all commands of an application are shown in this list. Commands are sorted alphabetically. Selecting the command shows the preset shortcut keys in the Current keys list.
Current keys	The list of keys assigned to the selected command.
Press new shortcut key	The field for entering a new shortcut key. Puts a human representation of any key pressed in the field. If the key pressed is already assigned to another command, the Currently Assigned To label appears with the current command name using that key.
Currently assigned to	Shows the name of the command for the new key entered when this key is already assigned to some other command. By default is hidden. This label is only visible if the entered key is already assigned to another command.
Assign	Assigns the key entered to the selected command. The key entered is added to the Current Keys list.
Remove	Removes the selected keys from the selected command.
Remove All	Removes all keys from the selected command.

Plugins pane

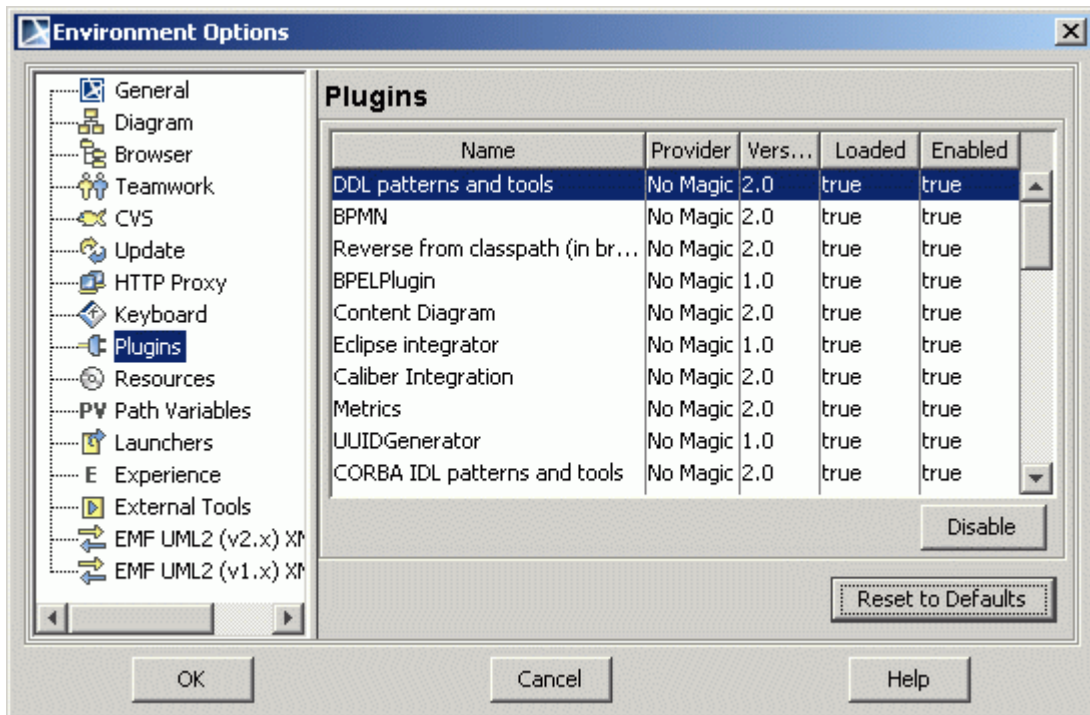


Figure 45 -- Environment Options dialog box. Plugins pane

Property name	Function
Name	The name of the plugin.
Provider	The name of the company which provides the plugin.
Version	Version number of the selected plugin.
Loaded	Indicates if the plugin is correctly loaded.
Enabled	Shows the status of the plugin. If True , the selected plugin is activated in MagicDraw.
Disable/Enable	Disables or enables the selected plugin.

Resources pane

Displays the external resources such as dtd, xsd, etc. that are used in MagicDraw.

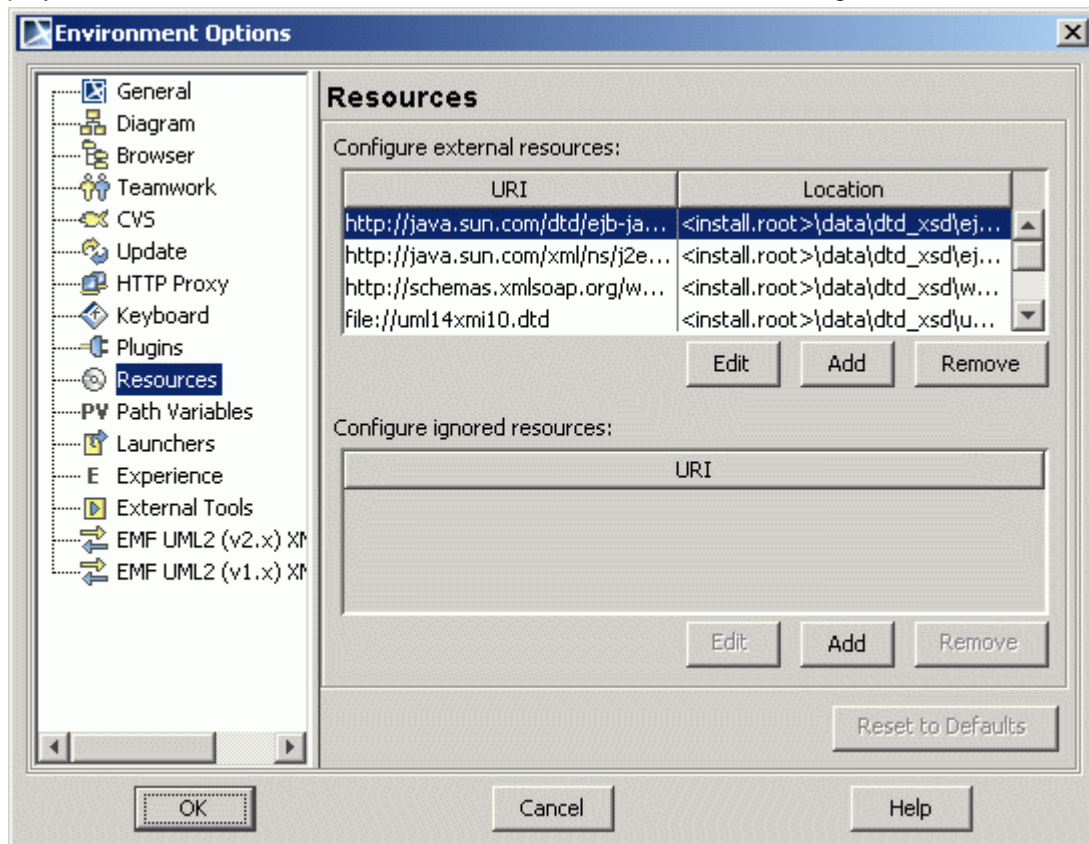


Figure 46 -- Environment Options dialog box. Resources pane

Property name	Function
Configure External Resources	Define external resources that will be used in XML parsing.
Location	The location of the external resource.
URI	The URI of the external resource.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Edit	Edit the location and the URI of the resource in the External Resource dialog box.
Add	Add a new external resource.
Remove	Remove the selected resource form the list.
Configure Ignored Resources	
URI	The URI of the external resource that will not be ignored in MagicDraw.
Edit	Edit the URI of the ignored external resource in the External Resource dialog box.
Add	Add a new external resource that will be ignored.
Remove	Remove the selected ignored resource from the list.

Path Variables Pane

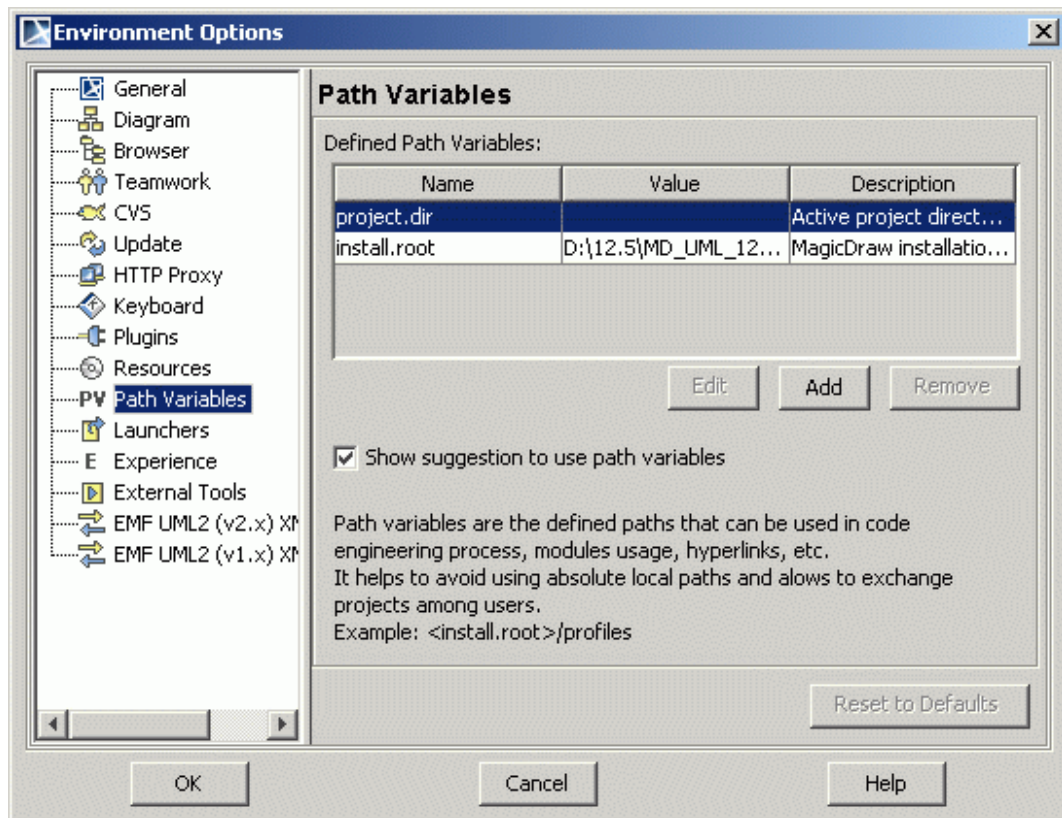


Figure 47 -- Environment Options dialog box. Path Variables pane

Property name	Function
Name	Any name for a path variable.
Value	Path to the folder.
Description	Description of the path variable.
Edit	In the list, select an existing path variable. Double click on the item in the list, or click the Edit button. The Path Variables dialog box opens. Edit the path variable.
Add	The Path Variables dialog box opens. Specify a name, value, and description for the new path variable.
Remove	Remove the selected path variable.

Property name	Function
Show suggestion to use path variables	In all places where path variables can be used, after specifying absolute path is suggested to use proper path variables. For this, the Use Path Variables dialog box will open.

Launchers Pane

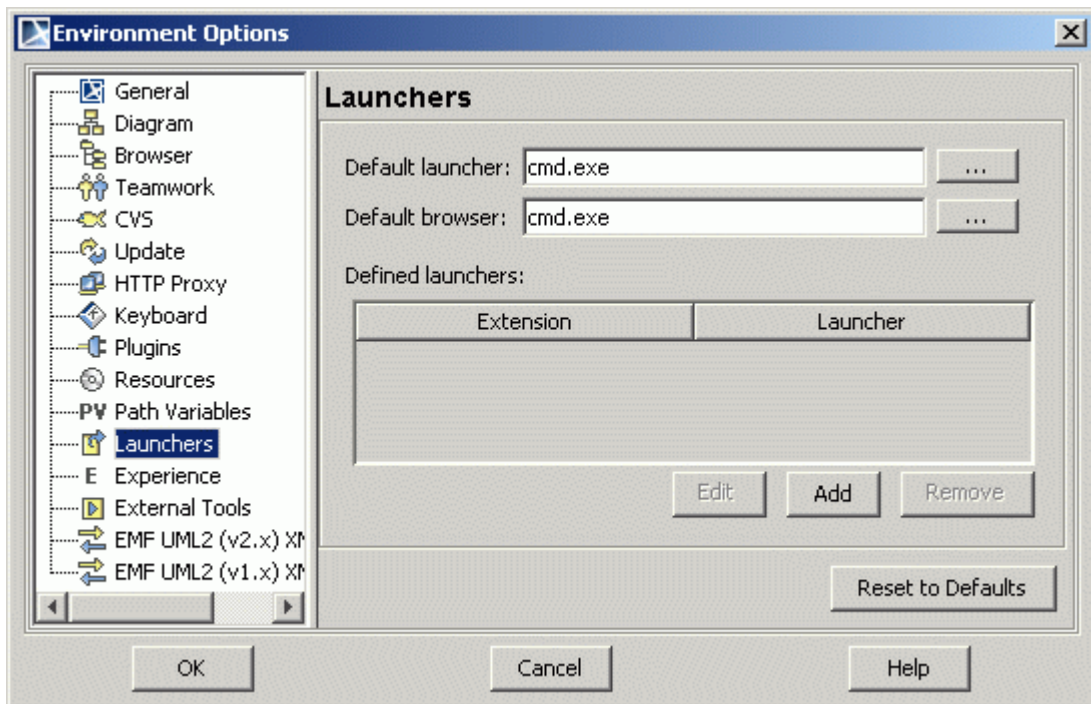


Figure 48 -- Environment Options dialog box. Launchers pane

Property name	Function
Default Launcher	The Launcher Properties dialog box opens.
Default Browser	The Launcher Properties dialog box opens.
Defined Launchers	
Edit	Opens the Launcher Properties dialog box for editing launcher information.
Add	Adds a launcher in the Launcher Properties dialog box.

Property name	Function
Remove	Removes a launcher from the list.

Experience Pane

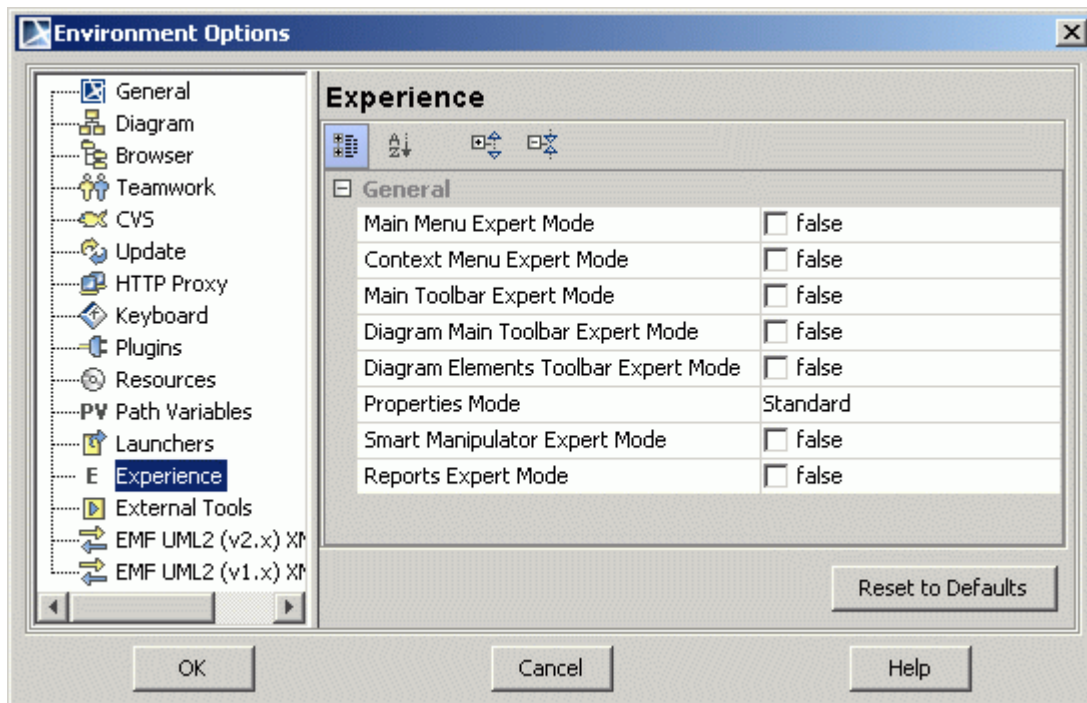


Figure 49 -- Environment Options dialog box. Experience pane

Property name	Function
Main Menu Expert Mode	Toggles visibility of the main menu by adding/removing arrows to expand menus.
Context Menu Expert Mode	Toggles visibility of the context menu actions by adding/hiding them, depending on which mode they were set to appear.
Main Toolbar Expert Mode	Toggles visibility of the toolbar menu buttons by adding/hiding them depending on which mode they were set to appear.
Diagram Main Toolbar Expert Mode	Toggles visibility of the diagram toolbar menu buttons by adding/hiding them depending on which mode they were set to appear.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Diagram Elements Toolbar Expert Mode	Toggles visibility of the diagram elements toolbar menu buttons by adding/hiding them depending on which mode they were set to appear.
Properties Mode	Possible choices: <ul style="list-style-type: none">• Standard• Expert• All
Smart Manipulator Expert Mode	Toggles visibility of buttons in the smart manipulators menu depending on which mode they were set to appear.
Reports Expert Mode	Toggles visibility of the report templates.

External Tools

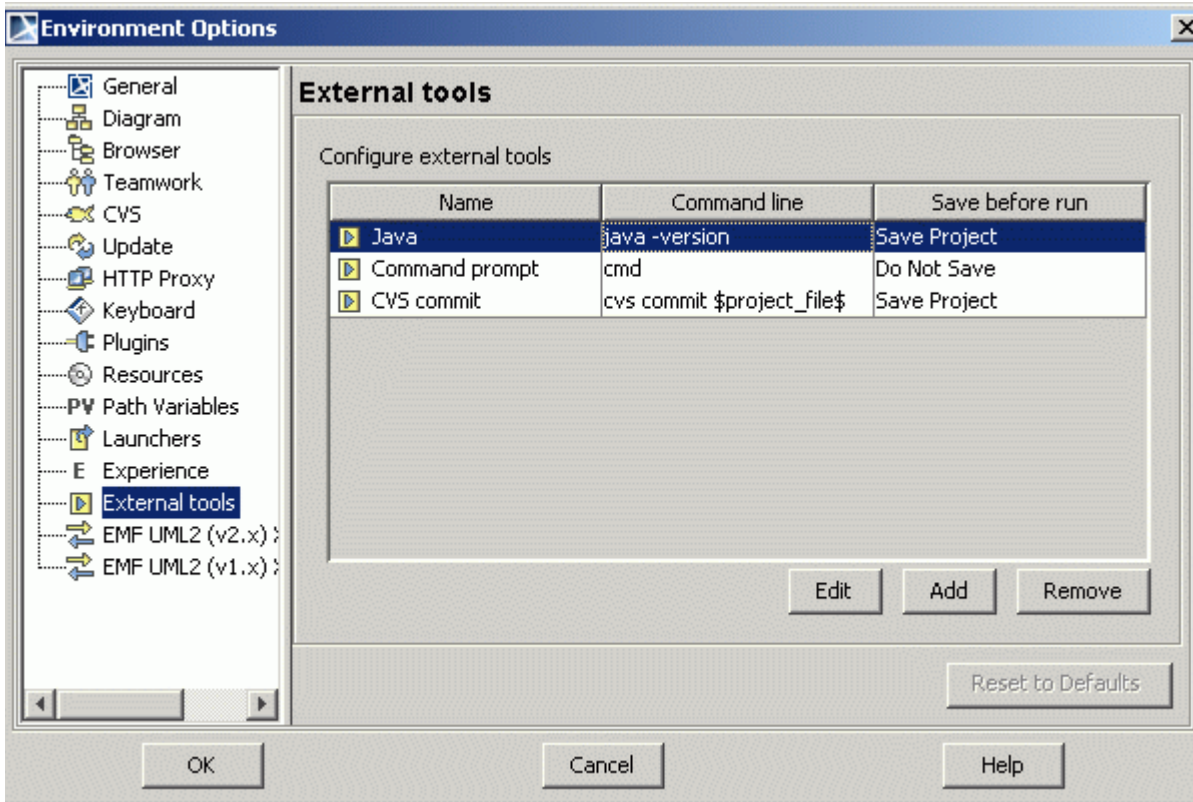


Figure 50 -- Environment Options dialog box. External tools pane

Property name	Function
Name	Name of the command, which will appear in the external toolbar.
Command line	The command line is executed upon selection. The command line can include the project file name variable, \$project_file\$, if it depends on the MagicDraw project file name.

Property name	Function
Save before run	Actions, which can be made with a project (save/export) before starting external tools that use a MagicDraw project file. Possible choices: <ul style="list-style-type: none">• Save project• Do not save• EMF UML2 (v1.x) XMI• EMF UML2 (v2.x) XMI
Edit	The External Tool dialog box opens. You may change the name, icon, command line text , and project save/export options here.
Add	The External Tool dialog box opens for new external tool configuration.
Remove	Removes the selected external tool from the list.

Scripts

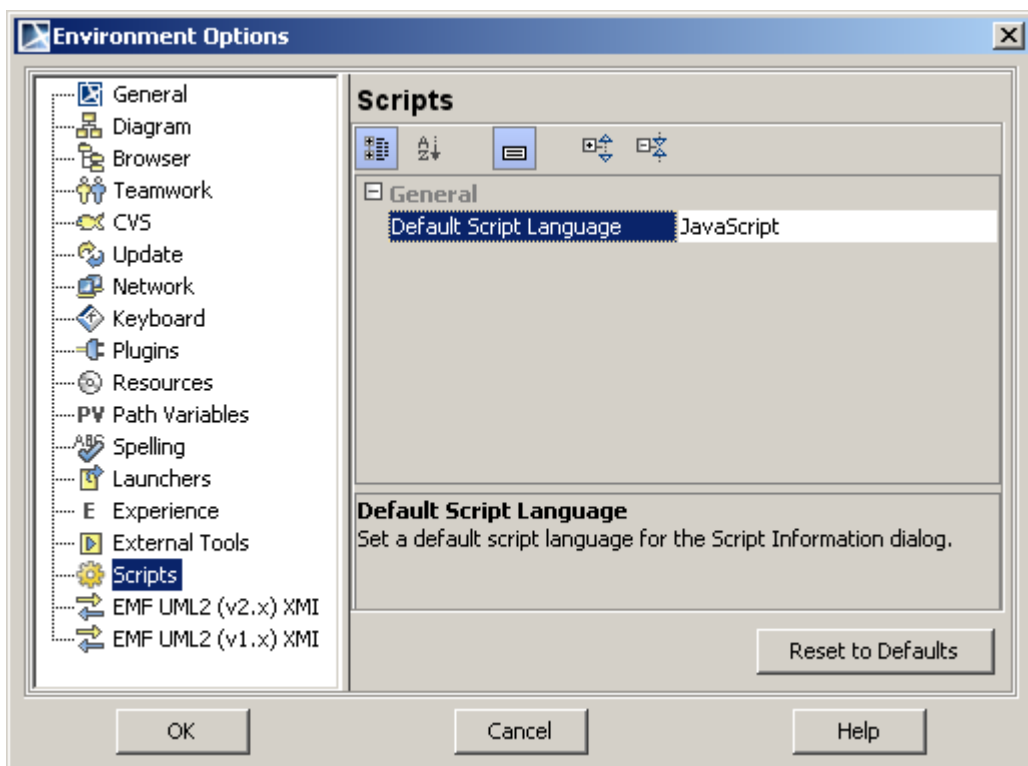


Figure 51 -- The Environment Options dialog box. Scripts pane

Property name	Function
Default Script Language	The default script language for the Script Engine.

For more information about Script Engine, see MagicDraw ScriptEngine UserGuide.pdf.

EMF UML2 (v2.x) XMI Options Pane

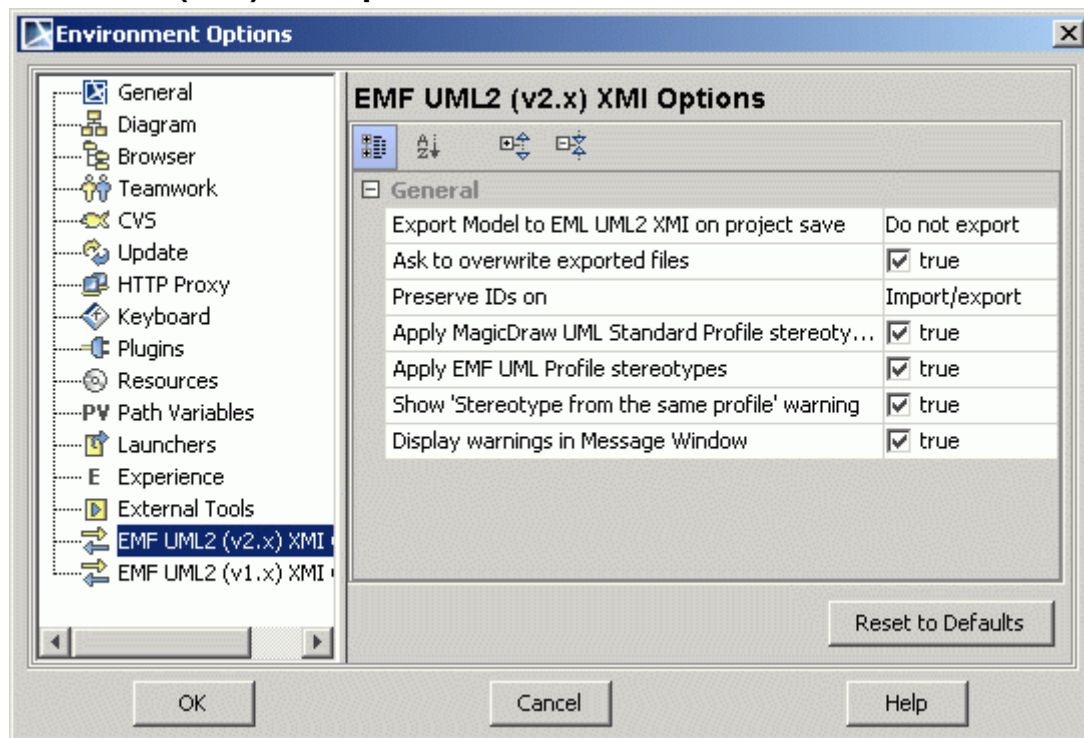


Figure 52 -- Environment Options dialog box. EMF UML2 (v2.x) XMI Options pane

Property name	Function
Export Saved Model to EMF UML2 XMI	Exports MagicDraw models (without diagrams) to UML2 format. Possible choices: <ul style="list-style-type: none"> • Do not Export • Ask Before Export • Always Export
Ask to overwrite exported files	If <i>true</i> , a message is displayed requesting confirmation to overwrite previously exported files.
Preserve Exported IDs	If selected, ID of the same exported element will not be changed when performing export several times.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Apply MagicDraw UML Standard Profile Stereotypes	MagicDraw contains standard UML stereotypes and EMF has standard UML stereotypes as well. If selected, exported elements will have stereotypes from the MagicDraw UML Standard Profile applied.
Apply EMF UML Profile Stereotypes	If selected, exported elements will have stereotypes from the EMF UML profile applied.
Show 'Stereotypes from the same profile' warning	If <i>true</i> , warnings that the stereotype will not be applied to the element, if this element is in the same profile with the stereotype, will appear when exporting.
Display warnings in Message Window	If selected, warnings related to project export, will be listed in the Message Window.

EMF UML2 (v1.x) XMI Options Pane

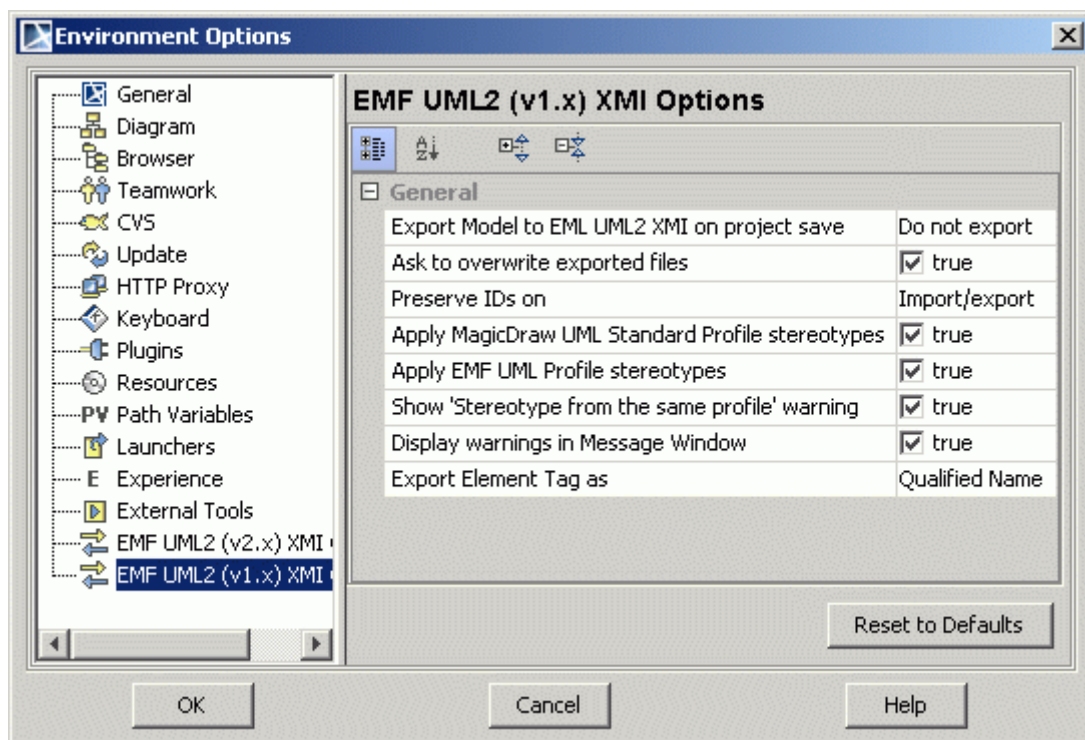


Figure 53 -- Environment Options dialog box. EMF UML2 (v1.x) XMI Options pane

Property name	Function
Export Saved Model to EMF UML2 XMI	Exports MagicDraw models (without diagrams) to UML2 format. Possible choices: <ul style="list-style-type: none"> • Do not Export • Ask Before Export • Always Export
Ask to overwrite exported files	If <i>true</i> , a message is displayed for requesting confirmation to overwrite previously exported files.
Preserve Exported IDs	If selected, ID of the same exported element will not be changed when performing export several times.

3 USING MAGICDRAW

Setting Environment Options

Property name	Function
Apply MagicDraw UML Standard Profile Stereotypes	MagicDraw contains standard UML stereotypes and EMF has standard UML stereotypes as well. If selected, exported elements will have stereotypes from the MagicDraw UML Standard Profile applied.
Apply EMF UML Profile Stereotypes	If selected, exported elements will have stereotypes from the EMF UML profile applied.
Show 'Stereotypes from the same profile' warning	If <i>true</i> , warnings that the stereotype will not be applied to element, if this element is in the same profile with the stereotype, will appear when exporting.
Display warnings in Message Window	If selected, warnings related to project export will be listed in the Message Window.
Export Element Tag as	Possible choices: <ul style="list-style-type: none">• Qualified Name• ID

Look and Feel: Controlling the Interface

The appearance of MagicDraw windows, dialog boxes, menus, and everything inside them can be changed.

To make changes to the interface:

From the **Options** menu, choose **Look and Feel** and then choose the style you wish.

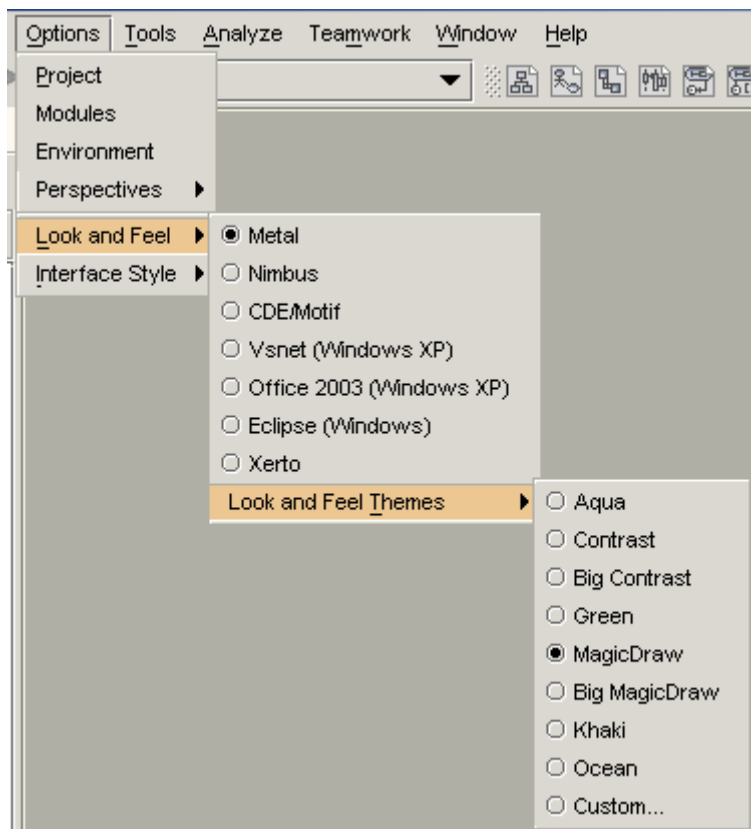


Figure 54 -- Look and Feel Themes submenu

The chosen style will not look exactly like the applications in those operating systems because every style of graphical interface is implemented within a Swing library, but it will look quite similar.

Depending on which operating system you use, some choices might be unavailable to you. For example, Windows9x/NT users may not switch to the Mac interface style.

Look and Feel Themes – themes listed in **Look and Feel Themes** are valid only for the metal style. Choose from any of the following themes:

- Aqua
- Contrast
- Big Contrast
- Green
- MagicDraw
- Big MagicDraw
- Khaki
- Ocean
- Custom – set your own options in the Properties dialog box.

Single and Multiple Windows interface styles

Beginning with MagicDraw version 7.5, the modern JIDE library is implemented (called Single Window interface style). Using the **JIDE** interface style, it is possible to work with the Browser window in a more flexible way, use documentation, zoom, and dock message windows above the main window. You can arrange the Browser window in combinations or even hide the desired Browser windows.

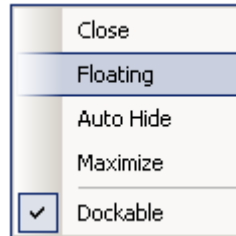
Also, you may use the different **Multiple Windows** style. It allows you to manage all windows independently, as if they belong to different applications. For instance, you can overlap the windows across each other, resize them independently, and so on. There is no main window containing all the other windows.

To set interface style

From the **Options** menu, select **Interface Style**, and then select one of the desired interface styles: **Single Window**, or **Multiple Windows**.

To make the Browser tab a separate window

From the Browser tab title shortcut menu, select **Floating**, and move the window to any desired position



If the **Dockable** check box is selected, the floating tab window will appear in a fixed edge position after trying to move it outside the MagicDraw window borders.

Assigning Shortcut Keys

To assign or change a command shortcut key

1. From the **Options** menu, select **Environment**.
2. The **Environment Options** dialog box opens.
3. Select the **Keyboard** pane and assign the desired shortcut keys in the right pane of the dialog box.

For more information about the **Keyboard** pane, see “Keyboard pane” on page 147.

4 WORKING WITH PROJECTS

The term “project” is used to describe the problem that must be solved, including all the possible solutions for how the problem can be resolved and finally developed. All work in MagicDraw™ UML is organized into projects. Project is the top entity where all model-related data (the set of diagrams) is held. Project data is organized by object orientation, which makes its management intuitive and in accordance with the problem that is being solved.

In this Section, you will find the following chapters:

1. "Creating a Project", on page 166.
2. "Saving a Project", on page 173
3. "Opening a Project", on page 175
4. "Importing a Project", on page 176
5. "Exporting a Project", on page 176
6. "Setting Project Options", on page 186
7. "Searching", on page 189
8. "Project Partitioning", on page 200
9. "Ecore Support" on page 231
10. "Working with Standard Profiles" on page 243

Creating a Project

[View Online Demo](#) MagicDraw Basics

Creating a new project

All project information is stored in a single file. A project name matches the file name where the project is saved.

The newly created project consists of the following packages:

- **Data** package is empty and holds all model elements.
- **File View** package contains components that are created during code engineering and represent source files. Adding a resident element to a particular component causes that element to be generated within the source file.
- **UML Standard Profile** contains stereotypes that are necessary for working with MagicDraw, primitive data types and constraints (which are UML standard), and UML 2 metamodel elements. The following data types are specified in MagicDraw: boolean, byte, char, date, double, float, int, Integer, long, short, void, and string.

You can also create your own packages for holding the model elements. By default, packages cannot be deleted or renamed in a project (except for the File View package).

To start a new project, you must create a new workspace for it.

To create a new workspace for a blank project

- From the **File** menu, select **New Project**.
- On the main toolbar, click the **New Project** button.
- Press shortcut key CTRL+N.

In all cases, the **New Project** dialog box opens.

1. Select the **Blank Project** icon.
2. Specify the file name in the **Name** text box.
3. Click the “...” button to select the location to store a newly created project in your computer. Click **OK**.

Working with multiple projects

Because you may need to manage several projects at the same time, MagicDraw allows you to work with several projects simultaneously.

All open projects are held in separate workspaces. Different active projects may exchange data. Entities from one project can be copied or moved to another.

To switch between loaded projects

- In the **Projects** drop-down combo box, click the additional project you wish to open.

4 WORKING WITH PROJECTS

Creating a Project

- Select **Projects** from the **File** menu, click the name of the project you wish to open.

To close all open projects

Select **Close All Projects** from the **File** menu. The **Question** message box appears.

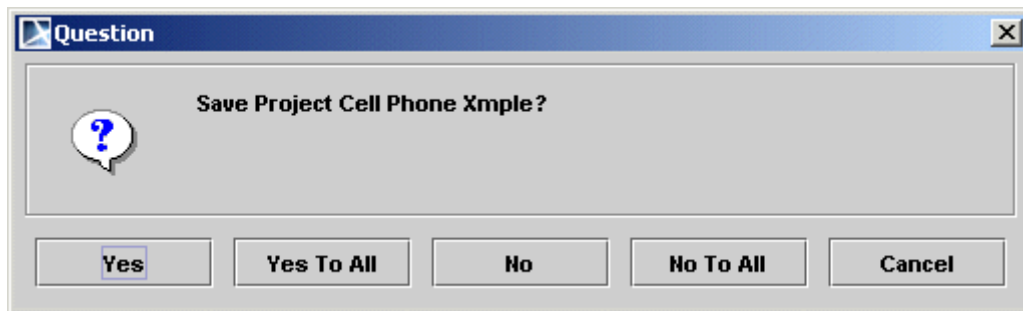


Figure 55 -- Question message box

Choose the way your projects will be closed:

Yes	The project you are currently closing will be saved (its name appears in the question). The dialog box is displayed again when the next project closes.
Yes To All	Save all projects without prompting. The Save dialog box will not appear for each open project.
No	Project you are currently closing will not be saved. The dialog box is displayed again when the next project closes.
No To All	All the projects will be closed without saving or further prompting.
Cancel	Cancel saving projects.

To exchange model entities between open projects

- Use the **Cut**, **Copy**, and **Paste** commands in the **Edit** menu, or the appropriate shortcut keys: Ctrl+X, Ctrl+C, Ctrl+V or the toolbar buttons.
- Drag-and-drop the created model element from the Browser tree to the Diagram pane.

NOTE Data may only be exchanged between projects that are currently open within MagicDraw. You may not copy/paste elements between instances of different tools that are currently running or to other applications.

Creating a new project from the existing source code

To create a new project from existing source files

- From the **File** menu, select **New Project**.
- On the main toolbar, click the **New Project** button.
- Press shortcut key CTRL+N.

In all cases, the **New Project** dialog box opens.

1. Select the **New Project from Existing Source** icon.
2. Specify the file name in the **Name** text box.
3. Click the "..."/>

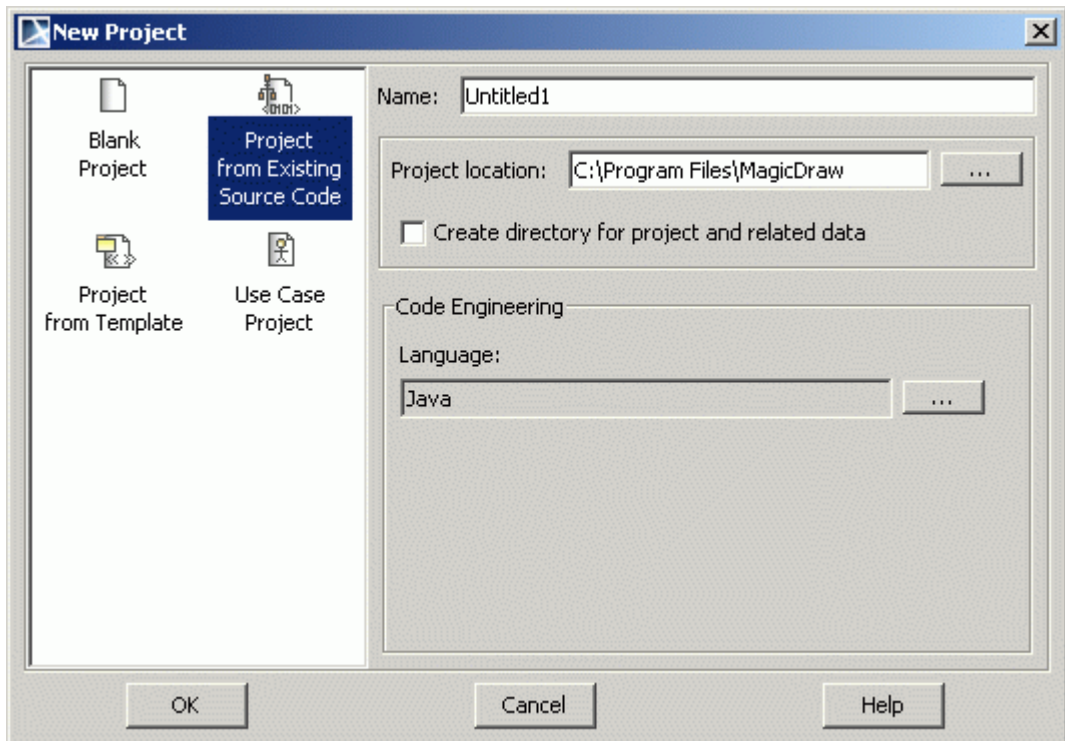


Figure 56 -- New Project dialog box - New Project from Existing Source

5. The **Round Trip Set** dialog box opens. Add the source files to enable code engineering to reverse them into a newly created project.

Creating a new project from a previously created template

NOTE: This functionality is available in Standard, Professional, Architect and Enterprise editions only.

C++, CIL, Java, C#, DDL, EJB, WSDL, XML Schema, Metamodeling, RUP extensions, CORBA IDL, and UMLWebExtension templates are available in the **New Project** dialog box.

To create a new project from a specified template

- From the **File** menu, select **New Project**.
- On the main toolbar, click the **New Project** button.
- Press shortcut key CTRL+N.

In all cases, the **New Project** dialog box opens.

1. Select the **New Project from Template** icon.
2. Specify the file name in the **Name** text box.
3. Click the “...” button to select the location to store a newly created project in your computer.

4 WORKING WITH PROJECTS

Creating a Project

4. Select the template from the templates tree and click **OK**.

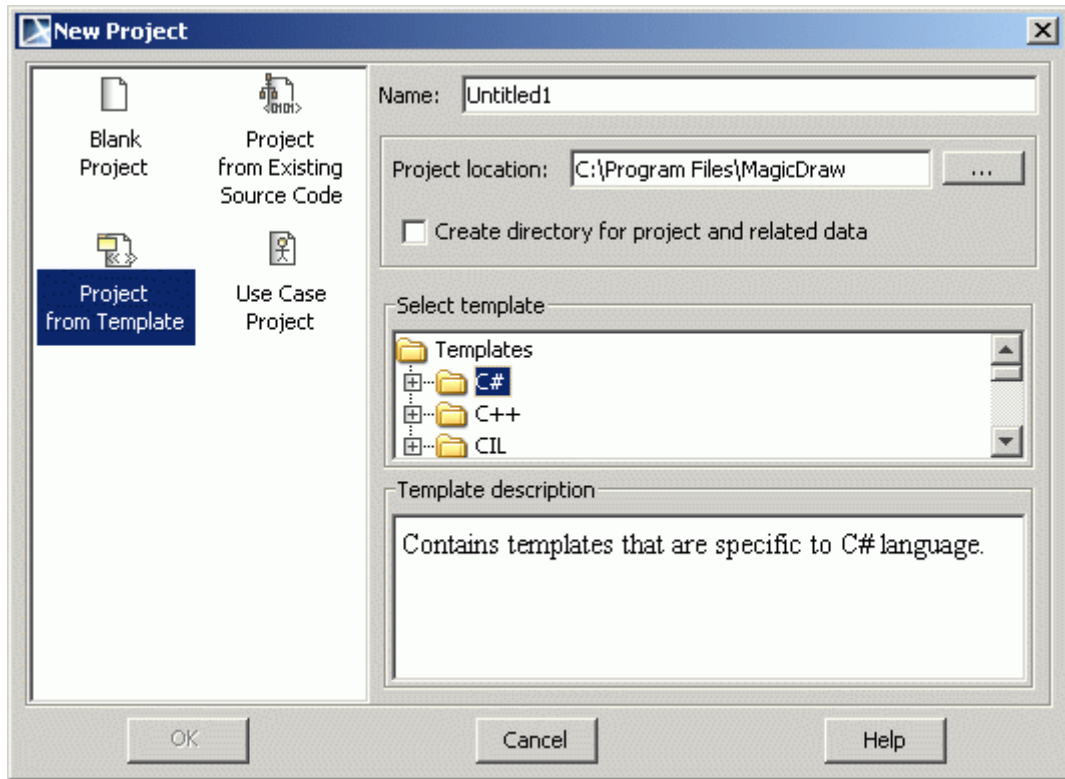


Figure 57 -- New Project dialog box - New Project from Template

The newly created project from a template will contain specific model elements and stereotypes.

TIP! All MagicDraw templates are located in the <MagicDraw installation directory>/templates folder so you can import the desired template into your previously created project using the **Import MagicDraw Project** command from the **File** menu.

Creating a new Use Case project

To create a new project from existing source files

- From the **File** menu, select **New Project**.

4 WORKING WITH PROJECTS

Creating a Project

- On the main toolbar, click the **New Project** button.
- Press shortcut key CTRL+N.

In all cases, the **New Project** dialog box opens.

1. Select the **Use Case Project** icon.
2. Specify the file name in the **Name** text box.

Click the “...” button to select the location to store a newly created project in your computer.

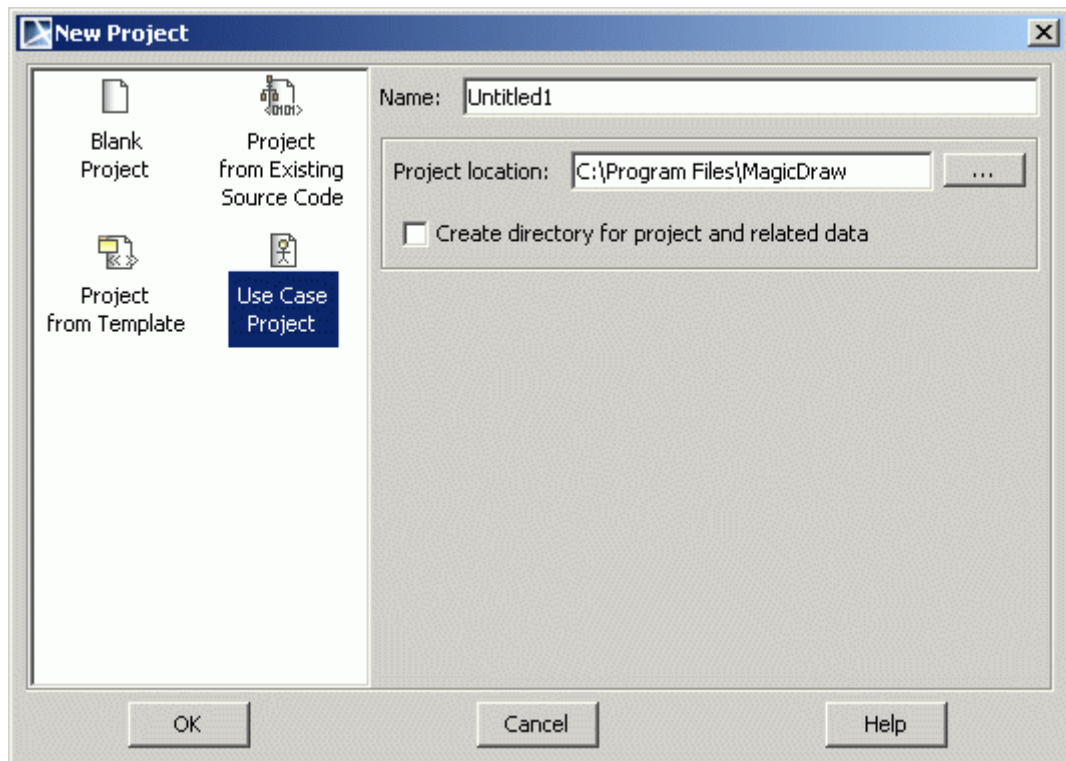


Figure 58 -- New Project dialog box - Use Case Project

The newly created project will automatically load the **UseCase Description Profile**. Also *Actor*, *High-Level Use Case* and *System-Level Use Case* packages will be created in the Data tree. Additional properties will be displayed in the newly created use cases **Specification** dialog box.

Saving a Project

IMPORTANT! The native MagicDraw format is *.mdzip and *.mdxml. Saving in *.xml, *.xml.zip format will also be allowed.

To save changes for later sessions, revised projects must be saved. While saving, you can edit the name of the project and the file format.

To save the project

1. From the **File** menu, select the **Save Project** or **Save Project As** command. Alternatively, you can click the **Save** button on the main toolbar or press the shortcut keys CTRL+S. The **Save** dialog box opens.
2. Select the destination directory (where you wish to save the project) and type the chosen file name.
3. Select the format for saving a project: **Packed MagicDraw File Format (*.mdzip)** (default) or **MagicDraw File Format (*.mdxml)**

4 WORKING WITH PROJECTS

Saving a Project

4. Select the version of **XMI 1.2**. Also, if you want to save maximum additional information to an xmi file (not required in loading to MagicDraw load, but may be useful when using other tools), select the **Rich XMI** button.

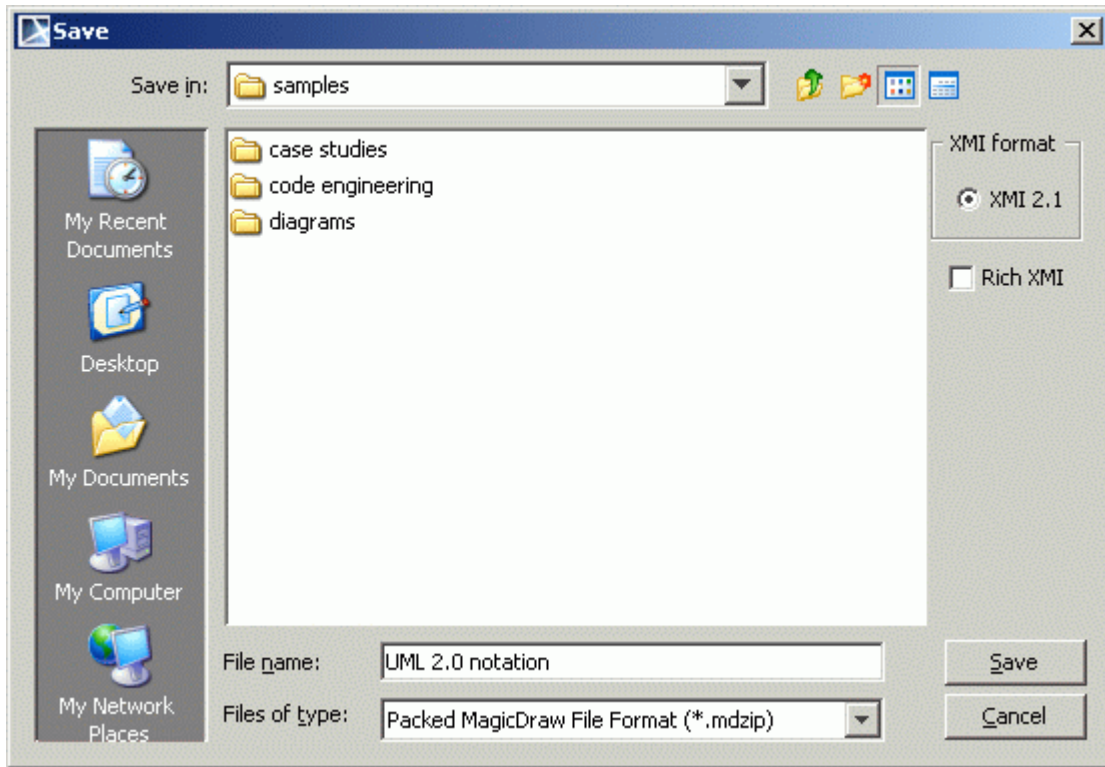


Figure 59 -- Save dialog box

NOTE

If the **Create Backup File** check box is selected in the **Environment Options** dialog box, MagicDraw always creates a backup file of the previously saved project. The backup is held in a file with a name identical to that of the project. For a detailed description of the **Environment Options** dialog box, see “Setting Environment Options” on page 129.

Autosave

After you stop working with MagicDraw, an idle time passes and the current project is saved to a special file called the AutoRecovery file.

If the application is terminated normally, the AutoRecovery file is removed. If the application crashes, the AutoRecovery file is left. On startup, MagicDraw checks for an AutoRecovery file. If it exists, MagicDraw suggests loading the project from this file.

To save an AutoRecovery file of the open project(s) when a system is not in use.

1. Open the **Environment Options** dialog box.
2. In the **General** pane, select the **Save Project Recovery Data on Idle** check box. Enter the system idle time (in minutes) in the **Idle Time to Activate Recovery Save** text box. This is the length of time the system must be idle in order to activate an AutoRecovery save.

Opening a Project

IMPORTANT! The native MagicDraw format is *.mdxml, or *.mdzip.

To edit or review previously created projects

1. From the **File** menu, select **Open Project** or click the **Open Project** button on the main toolbar.
2. In the **Open** dialog box, select the project you wish to open and click **Open**.

TIP: Drag-and-drop the project you wish to open directly from the open window to MagicDraw and the project is started immediately.

Double-click a project file with the *.mdxml or *.mdzip extension, it will open a new MagicDraw application window.

NOTES: If the **Open Last Project on Startup** check box in the **Environment Options** dialog box is selected, the last project you worked with will be opened right after MagicDraw starts.

XMI v2.1 is the native MagicDraw file format for model storage. But this format does not specify how to store diagrams. So, if you use MagicDraw to open an XMI file exported from another tool, only the model will be loaded (not diagrams or views). MagicDraw can not "import" XMI, it can only open it. For model interchange you should use the RConverter. For more about RConverter, See "Reusing model parts between models" on page 222.

You may load more than one project within the same MagicDraw session. A separate workspace will be created for each project that is opened.

Importing a Project

To import a previously created project to an open project

1. From the **File** menu, select **Import**.
2. The Import dialog box opens. Select the project you want to import and click OK. The diagrams of the imported project are placed in the open project.

Exporting a Project

Exporting a module of a project

You can export part of a project as a module and share it with other users/projects.

For more information about data partitioning, see “Project Partitioning” on page 200.

Exporting a project as a template

NOTE: This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

You can save (export) the created project as a template and use the same project for creating other new projects.

To export project as template

1. Open a project you want to export as a template. From the **File** menu, select **Export** and then **Template**.
2. The **Export Template** dialog box opens.
3. Type the name and a description of the template.

4. Click **OK**.

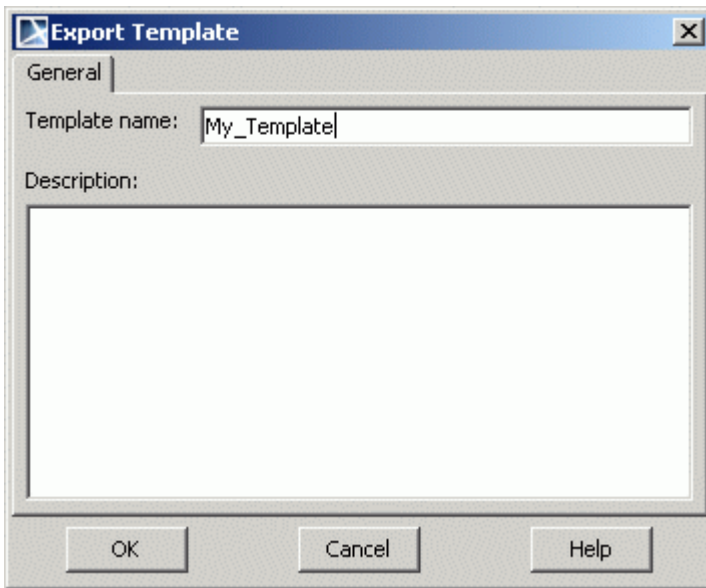


Figure 60 -- Export Template dialog box

Box	Function
Template Name	Type a template name.
Description	Type information about the template.
OK	Exports the template and the description to the Templates folder.
Cancel	Closes the dialog box without exporting the template.
Help	Displays MagicDraw Help

Exporting a project as MOF

This feature renews the MagicDraw metamodeling portfolio. MagicDraw is able to export/import the UML model into/from the MOF (both CMOF and EMOF) XMI format.

To export a project as a MOF

To export the model into the MOF, from the **File** main menu, select **Export** and then **MOF**.

To import a project as a MOF

To import the MOF from the **File** main menu, select **Open Project** and select to open **.emof* or **.cmof* file.

Data which can not be exported

- Diagram data can not be exported.
- None of the model features that are available only in UML will be exported (behavioral models in particular, etc).

MOF Domain model

Domain model for CMOF can be glanced at the MOF2.0 spec, 06-01-01.pdf document from OMG.

MOF export

Under File->Export->MOF group there are 2 items – MOF Whole Model, MOF Selection.

Exporting the whole MOF model

To export the whole model into the MOF, from the **File** main menu, select **Export**, **MOF**, and then **MOF Whole Model**. The **Select Target File** dialog box opens (see Figure 61 on page 179).

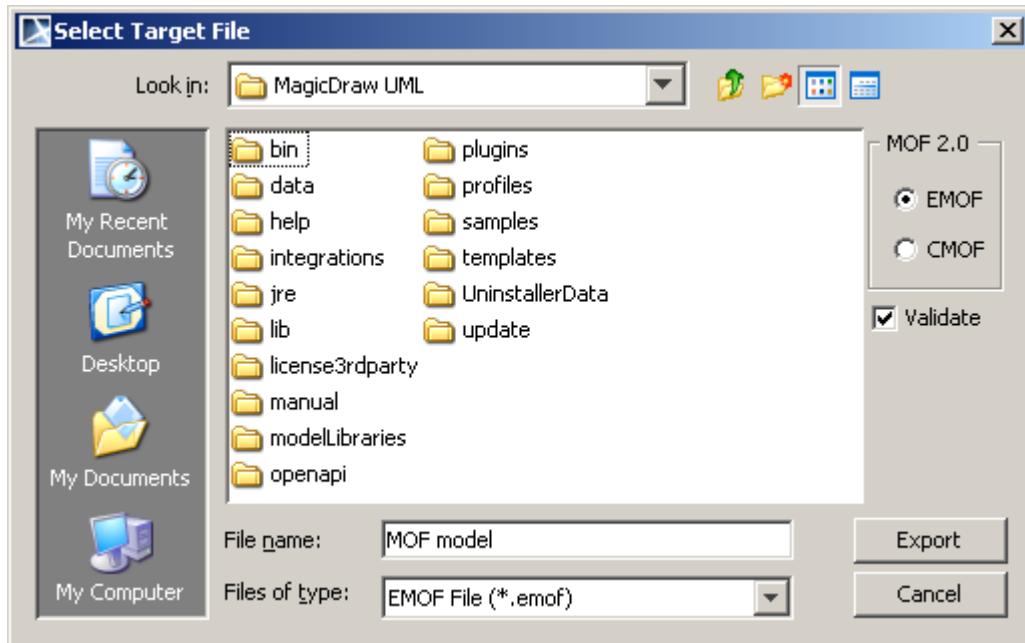


Figure 61 -- The Select Target File dialog box

Select directory to export the file, type the file name.

At the right side of dialog box you can choose to export EMOF or CMOF. The extension of file is by default .cmof or .emof correspondingly. After this the whole model is exported except the model parts marked as auxiliary resources (UML standard profile, Metamodeling profile, etc).

Select or clear the **Validate** check box for validating the MOF model export. When export runs, the exported model is also checked using the appropriate (CMOF/EMOF) validation suite. For more information about MOF model validation, see "Validation constraints" on page 180.

Exporting the selected part of the MOF model

To export the selected part of the model into the MOF, from the **File** main menu, select **Export, MOF**, and then **MOF Selection**. The **Select Packages to Export** dialog box opens (see Figure 62 on page 180). Select one or multiple packages to choose the model part to be exported. After this the selected model part (package granularity level) is exported to CMOF or EMOF correspondingly.

After the package selection, click OK and the Select Target File dialog box opens. Detailed description, see above “Exporting the whole MOF model” on page 178.

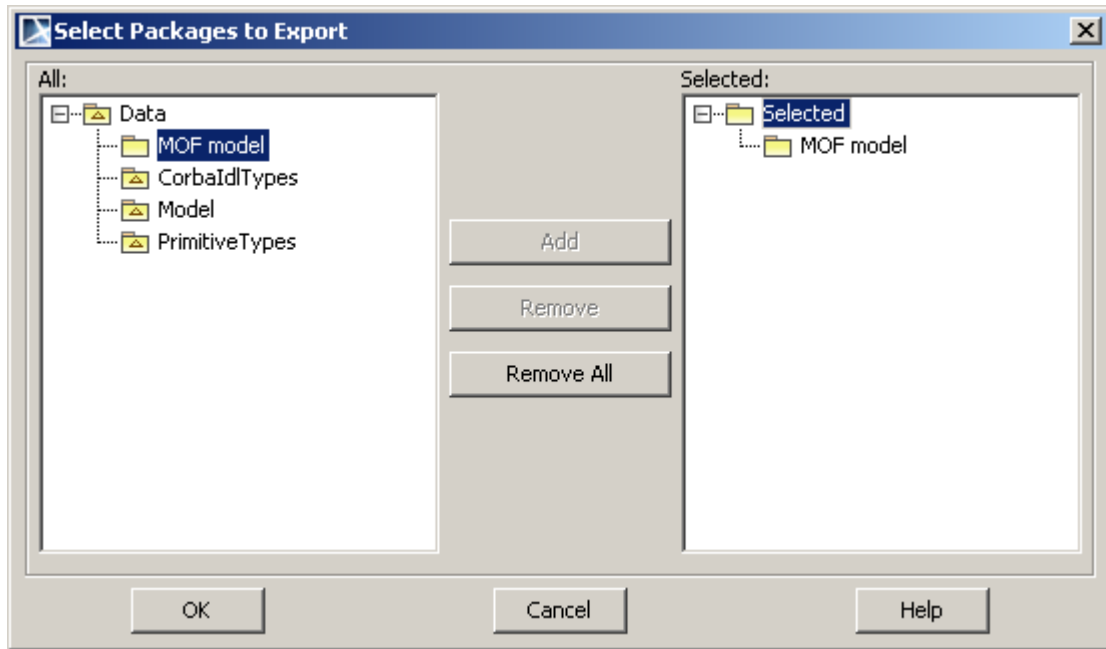


Figure 62 -- The Select Packages to Export dialog box

Metamodeling template

You can create a new project using a Metamodeling template. Template contains Metamodeling profile usage. It can be used as a starting point for developing your own metamodels. Also look at the Ecore Metamodel Example in the samples directory for a larger example of how to use the profile.

For more information about templates usage, see "Creating a new project from a previously created template", on page 170.

Validation constraints

There are 2 new validation suites for testing the model being exported to MOF (1 each for CMOF and EMOF). These suites contain batches of rules, which display warnings about elements not being exported. Rules are run together with model export (if the Validate checkbox is checked).

Rules are warnings only - they warn about elements not being exported (diagrams, behavioral elements etc) but do not preclude the model from being exported. Unsuitable elements are simply skipped.

Exporting a project as an EMF UML2 (v2.x) XMI

Export of a MagicDraw model to an EMF based UML2 (v2.x) compatible XMI. It allows the interchange of UML 2 models for further manipulations and transformations with the most popular MDA tools such as AndroMDA, OpenArchitectureWare, and other tools.

To export a project as an EMF UML2 XMI

1. Open a project you want to export as an EMF UML2 (v2.x) XMI. From the **File** menu, select **Export** and then **EMF UML2 (v2.x) XMI**.
2. The **Select Location** dialog box opens.
3. Type the name and select location for exporting the project.
4. Click **OK**. The project and profiles will be saved with the *.uml extension.

To change the export options

1. From the **Options** main menu, select **Environment**. The **Environment Options** dialog box opens.
2. Select the **EMF UML2 (v2.x) XMI Options** group and change the options in the right side pane. For more information, see “EMF UML2 (v2.x) XMI Options Pane” on page 159.

Exporting a project as an EMF UML2 (v1.x) XMI

Export of a MagicDraw model to an EMF based UML2 (v1.x) compatible XMI. It allows the interchange of UML 2 models for further manipulations and transformations with the most popular MDA tools such as AndroMDA, OpenArchitectureWare, and other tools.

To export a project as an EMF UML2 XMI

1. Open a project you want to export as an EMF UML2 (v1.x) XMI. From the **File** menu, select **Export** and then **EMF UML2 (v1.x) XMI**.
2. The **Select Location** dialog box opens.
3. Type the name and select a location for exporting the project.

4. Click **OK**. The project and profiles will be saved in *.uml2 extension.

To change the export options

1. From the **Options** main menu, select **Environment**. The **Environment Options** dialog box opens.
2. Select the **EMF UML2 (v1.x) XMI Options** group and change the options in the right side pane. For more information, see “EMF UML2 (v1.x) XMI Options Pane” on page 161.

Exporting to BPEL

The Business Process Execution Language (BPEL) is an XML schema language for specifying the automated business processes bidirectionally between multiple enterprises using Web Service technology. The BPEL is suitable for all technical staff to manipulate data of process and execute the business process.

The MagicDraw UML BPEL Export exports the Business Process diagram to the valuable BPEL, based on BPEL specification v.1.1.

The Business Process Diagram (BPD) is a helpful tool for business analysts/users to generate a model for business processes. Different from other UML diagrams such as an Activity Diagram or a Use Case Diagram, the BP diagram allows technical staff to easily understand and implement business diagrams. To optimize the valuation of BPD, converting BPD to the execution form (BPEL), which can use the benefit of Web Service, is quite a challenge.

To export a BP diagram to BPEL

1. Create Business Process diagram.
2. From the **File** menu, select **Export** and then **BPEL**. The **BPEL Export** dialog box opens.
3. Select the existing Business Process diagram in the tree and click the **Export** button. The **Save** dialog box opens.
4. Save the BPEL file in the selected location. A message stating the export was successful will appear.

If the BPEL export can not be performed, a Message Window appears with a detailed description of problems.

To evaluate validation rules for export

After exporting the BPMN notations from the BPD diagram to the BPEL files, some additional process data and Web Services information require manipulation in order to deploy the BPEL files on the vendor-specific servers.

It is because BPEL version 1.1 is not a standard yet. There are still many open issues and questions about implementation. Each vendor has made their own decision on specific implementation on the open issues and those questions. BEA is the chosen vendor of this version of MagicDraw BPEL Export. This MagicDraw BPEL Export version will focus on the BEA Web Logic Workshop (WLW) and the BEA Web Logic Integration (WLI); therefore the limitations and open issues from BEA are concerned.

In order to export a BP Diagram based on BPMN specification version 1.0 to a BPEL file based on BPEL4WS Version 1.1 specification with a flavor of BEA, the validation rules for mapping between two standards are required and can be found below.

Ignored Notations

Note, Note HTML, Text Box, Text Box HTML, Anchor to Note, Constraint, Separator, Rectangular-Shape and Group will be ignored and will not be mapped.

UnMappable Notations

Start Event None, Intermediate Event None, Intermediate Event Link, Intermediate Event Cancel, Intermediate Event Multiple, End Event None, End Event Cancel, End Event Multiple, Compensation Task, Gateway Complex, Manual Task, Reference Task, Expanded Sub-Process Compensation, Collapsed Sub-Process Compensation, Collapsed Ad-Hoc Sub-Process, Expanded Ad-Hoc Sub-Process, Association, Directional Association, Compensation Association, Message Flow, Data Object, Horizontal Pool, and Vertical Pool cannot be mapped.

Event Notations

Start Event

- A process level must have one and only one Start Event notation.
- Start Event notation must be a source of Sequence Flow. This means the Start Event notation must have an outgoing Sequence Flow.

End Event

- At least one End Event notation must be present in the diagram.

- Multiple End Events are allowed in the diagram.
- Every branch of the process must have the End Event notation. However, in Expanded Sub-Process notation, the End Event None notation can be used as the ending point.

Intermediate Event

- Intermediate Event notation must be between Start Event and End Event notations.
- Only Message, Timer, Error, and Rule Intermediate Event can be used in normal flow.
- Intermediate Message, Timer, Error, and Rule must have only one incoming Flow.
- Intermediate Event must have only one outgoing Sequence Flow.
- Intermediate Event Link will not be mapped because the methodology of mapping needs to track the flow to the element. Therefore, if the Intermediate Event Link does not have a flow to it, it will not be recognized.

Task and Sub-Process

- Task and Sub-Process cannot have any Event attached to them.
- Task notations must have only one incoming Sequence Flow. In case there are multiple incoming Sequence Flows for a Task, it is recommended to use the Gateway for convergence, instead.
- A diagram (process) inside the Expanded Sub-Process must follow the same rules as the outer diagram (process).

An exception to this is the End Event None can be used in the Expanded Sub-Process.

All notations inside the Expanded Sub-Process cannot have an incoming Sequence Flow or an outgoing Sequence Flow linked directly to notations outside the Expanded Sub-Process.

Gateway

- If a Gateway has only one incoming and one outgoing Sequence Flow, the gateway is not needed in the BP Diagram.
- A Gateway cannot have both multiple incoming Sequence Flows and multiple outgoing Sequence Flows. If it has multiple incoming Sequence Flows, it must have only one outgoing Sequence Flow. If it has multiple outgoing Sequence Flows, it must have only one incoming Sequence Flow.

DataBased XOR Gateway

- If there is only one incoming Sequence Flow to the Data-based XOR Gateway and more than one outgoing Sequence Flow, the Data-based XOR Gateway will act as a convergence. A Data-based XOR Gateway must have one outgoing Default Sequence Flow and at least one

outgoing Condition Sequence Flow with condition expression. The normal Sequence Flow cannot be an outgoing Flow from a Data-based XOR, which acts as a convergence.

- If there are more than one incoming Sequence Flows to a Data-based XOR Gateway and only one outgoing Sequence Flow, the Data-based XOR Gateway will act as a divergence. The only outgoing Sequence Flow from a Data-based XOR Gateway must be a normal Sequence Flow and not a Condition Sequence Flow or Default Sequence Flow.

EventBased XOR

- For an EventBased XOR Gateway, the outgoing Sequence Flow can only target a Task, with the TaskType set to Receive or an Intermediate Event with a Trigger attribute set to Message, Timer, or Rule.
- If there is only one incoming Sequence Flow to an Event-based XOR Gateway and more than one outgoing Sequence Flow, the Event-based XOR Gateway will act as a convergence. The only outgoing Sequence Flows from an Event-based XOR Gateway must be a normal Sequence Flows and not a Condition Sequence Flow or Default Sequence Flow.
- If there are more than one incoming Sequence Flows to an Event-based XOR Gateway and only one outgoing Sequence Flow, the Event-based XOR Gateway will act as a divergence. The only outgoing Sequence Flow from an Event-based XOR Gateway must be a normal Sequence Flow and not a Condition Sequence Flow or Default Sequence Flow.
- If the outgoing Sequence Flow targets a Receive Task, then an Intermediate Event with a Trigger Message cannot be a target of the other outgoing Sequence Flow of this Event-based XOR Gateway.

Inclusive OR

- If there is only one incoming Sequence Flow to an Inclusive OR Gateway and more than one outgoing Sequence Flow, the Inclusive OR Gateway will act as a convergence. An Inclusive OR Gateway must have one outgoing Default Sequence Flow and at least one outgoing Conditional Sequence Flow with condition expression. The normal Sequence Flow cannot be an outgoing Flow from an Inclusive OR, which acts as a convergence.
- If there are more than one incoming Sequence Flows to an Inclusive OR Gateway and only one outgoing Sequence Flow, the Inclusive OR Gateway will act as a divergence. The only outgoing Sequence Flow from an Inclusive OR Gateway must be a normal Sequence Flow and not a Condition Sequence Flow or Default Sequence Flow.

Parallel AND

- If there is only one incoming Sequence Flow to a Parallel AND Gateway and more than one outgoing Sequence Flow, the Parallel AND Gateway will act as a convergence (Parallel AND

Fork). The only outgoing Sequence Flows from a Parallel AND Gateway must be a normal Sequence Flows and not a Condition Sequence Flow or Default Sequence Flow.

- If there are more than one incoming Sequence Flows to a Parallel AND Gateway and only one outgoing Sequence Flow, the Parallel AND Gateway will act as a divergence (Parallel AND Join). The only outgoing Sequence Flow from a Parallel AND Gateway must be a normal Sequence Flow and not a Condition Sequence Flow or Default Sequence Flow.

Sequence Flow

- All notations, except Start Event must have an incoming Sequence Flow
- All notations, except End Event, must have an outgoing Sequence Flow.
- Isolate notation is prohibited in the diagram. This means that any notations without any incoming and outgoing flow are not allowed in the diagram.
- All notations, except Gateway, must not have more than one incoming Sequence Flow.
- Default Sequence Flow and Condition Sequence Flow are only allowed to be used with the Gateway notation.

Setting Project Options

Project Options

The **Project Options** dialog box is used for defining properties for the model elements (shapes and paths) and diagrams, for creating your own project style, for importing or exporting a created project style, for applying the default element properties, and for defining code engineering options.

4 WORKING WITH PROJECTS

Setting Project Options

To define Project Options

From the **Options** main menu, select **Project**. The **Project Options** dialog box opens.

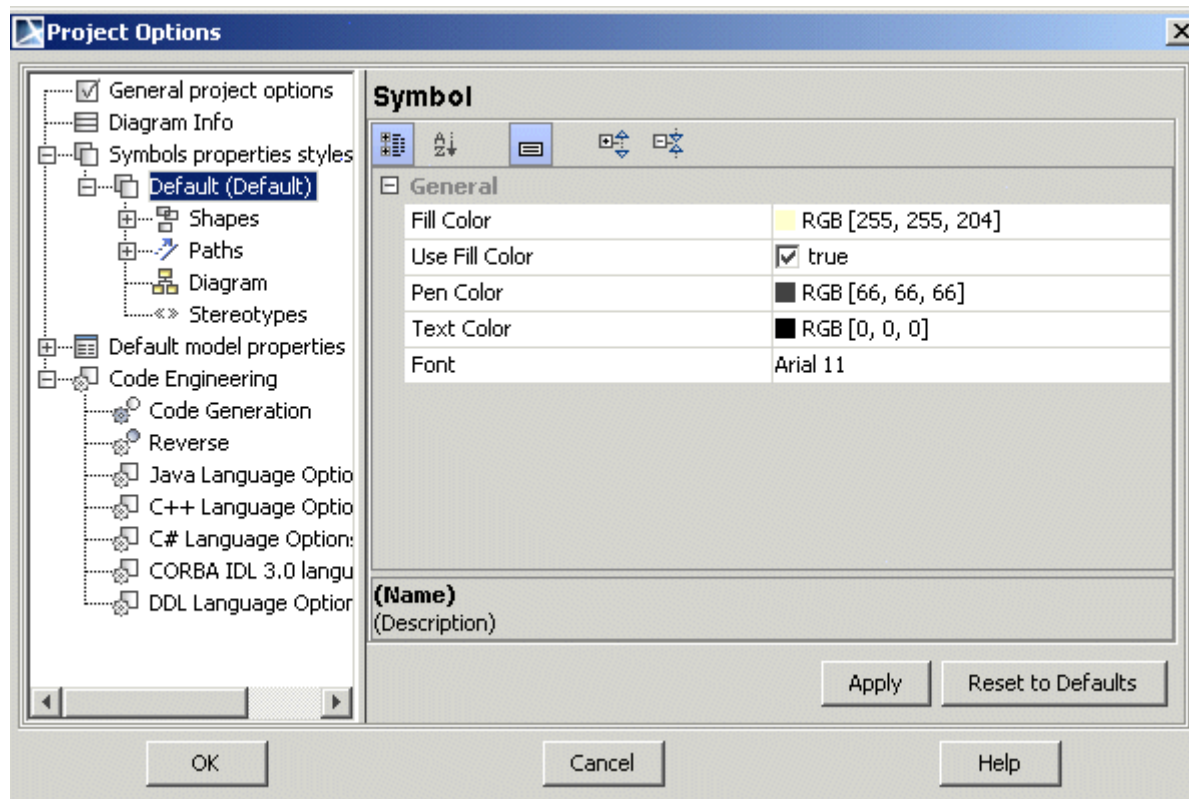


Figure 63 -- Project Options dialog box

The **Project Options** dialog box has several collections of customizable options:

General Project Options	<ul style="list-style-type: none"> • In the Modules Path text box, enter the path where by default modules will be stored. Define the default packages where the extension elements will be stored. Also, if desired, you can select the package every time when creating a new extension element, the dialog box asking about the storage place will open. In the Modules Path text box, enter the path where, by default, modules will be stored. • If the Auto synchronize Parameters and Arguments check box is selected, arguments will be created on parameters creation and they will be synchronized. After Auto synchronize Parameters and Arguments check box is cleared parameters and arguments are not synchronized anymore. By default the Auto synchronize Parameters and Arguments check box is selected. For more information about parameters and arguments synchronization see "Parameters synchronization with Arguments" on page 960. • The Qualified name display style option: <ul style="list-style-type: none"> - Absolute. - Model Library relative (Model Library is a package with <i>modelLibrary</i> stereotype). - Model relative means that element's qualified name shown on its symbol (not property) should be cropped from the nearest model found in hierarchy without its name. - Model or model library relative option combines two options "Model relative" and "Model Library relative" in one option.
Diagram Info	Customize how and what information will be displayed in the Diagram Info table.
Symbols Properties Styles	<p>Expands the tree hierarchy of all the styles defined within the project. You can create as many of these styles as you like.</p> <p>NOTE To apply the changed project options to an open project, click the Apply button.</p> <p>For more information about the style engine, see "Style Engine" on page 346.</p>

Default model properties	<p>You may define model element properties in the Project Options dialog box, in the Default model properties branch. Select the exact element in the tree and change the property value in the right side pane. To reset the element properties to the default value, click the Reset to Defaults button. To reset the property values for all elements select the Default model properties branch and click the Reset to Defaults button.</p> <p>For more information about setting the default element property values, see “Default Property Values” on page 339.</p>
Code Engineering	<p>Define general code engineering options, as well as options specific to languages. For more information, see Code Engineering user's guide.</p>

To find elements to change, browse the options tree in the **Project Options** dialog box. The items in this hierarchy are either:

- **Compressed** - a plus sign next to an icon indicates that the icon is compressed, and contains other model elements. This is the default setting when you start your application. Click the plus sign to expand the icon and view its subordinate items.
- **Expanded** - a minus sign next to an icon indicates that the icon is fully expanded. Click the minus sign to collapse the item.

If there is no sign next to an icon, it does not contain other model elements.

Searching

The MagicDraw search mechanism enables searching within model elements data, symbols, and extensions.

You may also search for usages and dependant elements of the selected elements. This functionality is described in the “Tooltip text” on page 279.

To quickly find the needed classifier or diagram

From the **Edit** menu, select **Quick Find**. In the dialog box that appears, type the name of the classifier or diagram (also, you can select it from the drop-down list) and choose one of the option buttons.

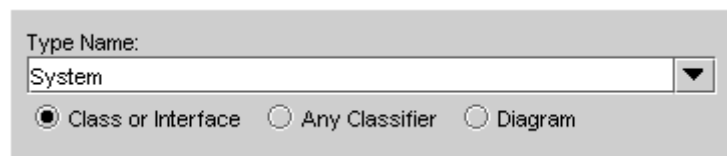


Figure 1 -- Quick Find screen

Filters in the autocompletion dialog allow the filtering of rarely used items, such as "metaclasses" and "elements from modules". This allows comfortable and clear usage of the autocompletion dialog for modeling, without active usage of the elements from profiles and it increases modeling speed.

At the bottom of the drop-down list box, you will find buttons to perform filtering:

- **The Auto completion includes metaclasses button.** When pressed, the list of available elements, element types, or stereotypes includes metaclasses (in MagicDraw metaclasses are placed in the *UML Standard Profile*) appears.
- **The Auto completion includes elements from profiles and modules button.** When pressed, the list of available elements, element types or stereotypes includes elements, which are placed in modules appears. (**Note!** This option toggles all profiles except the *UML Standard Profile*.)
- **The Auto completion uses camel case button.** When pressed, you may search for elements via the capital letter patterns. For example, instead of typing *ArrayIndexOutOfBoundsException* you may type *AIOOBE*.

Find all elements of the same type

1. From the **Edit** menu, select **Find**, or press corresponding shortcut key Ctrl+F, or click the **Find** button on the main toolbar. The **Find** dialog box opens.
2. Type the "*" symbol in the **Name** text box.
3. Click the "..." button near the **Type** text box to open the **Select Element/Symbol Type** drop-down combo box. Select the types of elements and click **OK**. The Model elements in this box are listed according to the metamodel.

4. Click the **Find** button to start search. The search results will appear in the **Search Tree** in the Browser.

TIP!

To generalize the beginning or ending of the name, add the "?" symbol to the front or to the end of the string.

Find model elements and symbols in your project

1. Choose **Find** from the **Edit** menu or press the corresponding shortcut key Ctrl+F or click the **Search** tab in the Browser.
2. In the **Name** text box, type the name of the element. If you want to find all elements of the selected type, enter the "*" symbol in the **Name** text box.
3. Click the "..." button near the **Type** text box to open the **Select Model Element/Symbol Type** drop-down combo box. Select the types of elements and click **OK**. The Model elements in this box are listed according to the metamodel.
4. To start a search, click the **Find** button. The search results will appear in the **Search Results** tree in the Browser.

TIP!

To generalize the beginning or ending of the name, add the "?" symbol to the front or to the end of the string.

NOTE

If the **Clear Previous Results** check box is cleared, new results are appended to the previous search results in the tree.

TIP!

Select **Search Data Unused in Diagrams** check box to find only elements without shapes.

To search for symbols in an active diagram

Search in the active diagram using the **Find** or **Quick find** dialog is a time saving feature.

You can search for the symbols of elements, which are drawn in the open diagram:

1. Select the **Find** in Diagram command from the diagram shortcut menu. The **Find** dialog box opens. In the **Find** dialog box, the **Limit results to active diagram** check box is selected.
2. Type the element name for the symbol you are searching for. Click the **Find** button.
3. In the **Search Results** tree, double click on the element and the symbol of this element is selected on the diagram pane.

-or-

- Press the **Shift+F** key. The **Find** dialog box opens.
- Press the **Ctrl+Shift+F** key to open the **Quick find** dialog.

Find dialog box

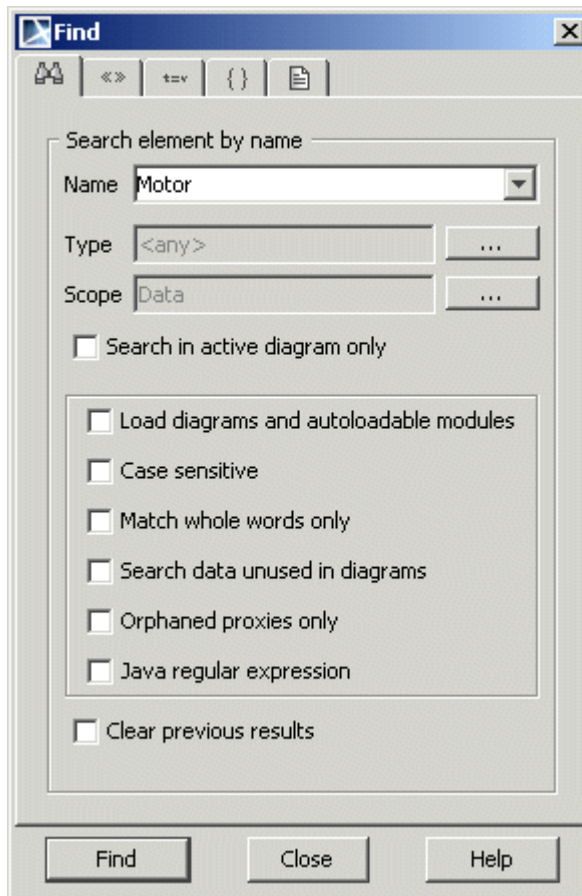




Figure 64 -- Find dialog box

The **Find** dialog box contains five tabs:

Tab Name	Tab Icon
Search Element by Name	
Search Element by Stereotype	

4 WORKING WITH PROJECTS

Searching

Tab Name	Tab Icon
Search Element by Tagged Value	t=v
Search Element by Constraint Value	{ }
Search Element by Documentation	

Search Elements

Element	Function
Name	<p>Type the name of the item you wish to find.</p> <p>NOTE: You may define wildcards <*> and <?> for the search. For example, if you define the following input string <a*b>, the system looks for items with <a> at the beginning and at the end of the string. If you define the string as <?agicDraw>, all strings containing <agicDraw> will be found.</p>
Type	Select an element type from the proposed items, or leave the default value of <any>.
Scope	Specify a package where the content search will be performed.
Limit results to active diagram	<p>The search scope is limited to the symbols of active diagrams. The Check box is disabled if all diagrams are closed or if the active diagram is empty. The default value is <i>false</i>.</p> <p>If the Find dialog box is opened from the diagram shortcut menu, the Limit results to active diagrams check box is selected.</p> <p>NOTE: The Limit results to active diagrams check box does not exist in the Find dialog box when it is opened from the model comparing dialog.</p>
Value	<p>Select or input a value of the Tagged Value or Constraint Value.</p> <p>NOTE: Only available for Search Element by Tagged Value and Constraint Value tab.</p>
Load elements (not loaded) and autoloadable modules	<p>If the model has diagrams or modules that are not loaded, select this check box to load all elements to be included in the search.</p> <p>NOTE: Elements will not be included in the search if the module load mode is set to Manual load.</p>
Case Sensitive	<p>Search for items that have capitalization exactly as defined in the string entered in the Item to Find box.</p> <p>When the check box is cleared, MagicDraw does not distinguish between uppercase and lowercase characters of the item name entered in the Name box while searching.</p>
Match Whole Words Only	<p>Search for items with names that exactly match the string entered in the Item to Find box.</p> <p>When the check box is cleared, MagicDraw searches for items with names matching the first part of the string entered in the Name field.</p>

Search Data Unused in Diagrams	Only searches elements that do not contain symbols in any diagram.
Java Regular Expression	In the “Java Regular Expressions” on page 195, you can find several expressions that will help you to make a search in MagicDraw.
Clear Previous Results	Removes all previous search results from the Browser tree.
Find	Searches for items and displays the results in the found items list field. If MagicDraw does not find any items, a message is displayed.
Close	Exits the dialog box.
Help	Displays MagicDraw Help .

Java Regular Expressions

Metacharacters

There are several characters supported, which are used to form search patterns $([\{\^{\$}\})?^{*+}$.

There are two ways to force a metacharacter to be treated as an ordinary character:

- + Precede the metacharacter with a backslash
- + Enclose it within \Q (starts the quote) and \E (ends it).

Character Sets

[abc]	Any character of a, b, or c.
[^abc]	Any character except a, b, or c (negation).
[a-z]	All characters from a to z (range).
[a-z[A-Z]]	All characters from a to z and A to Z (union).
[a-z&&[r-z]]	Characters from r to z (intersection).
[a-z&&[^r-z]]	Characters from a to q (subtraction).

Predefined character sets:

.	Any character.
\d	Any digit character.

\D	Any non digit character.
\s	White space character (\t\n\x0B\f\r).
\S	Any non white character.
\w	Word character (a-z, A-Z,_,0-9).
\W	Any non word character.

Example:

Regular expression: `[ABC][^\s]\d`

Matched text: any sequence starting with an "A", "B", or "C" symbol, followed by any non white space character and any digit.

Grouping

Capturing groups helps to treat multiple characters as a single unit.

Example:

Regular Expression: `ABC|(\dABC)`

Matched text: any text containing ABC symbol set or ABC symbol set beginning with any digit symbol.

Quantifiers

Quantifiers allow specify a number of character (X) appearances.

X?	Match X zero or one time.
X*	Match X zero or many time.
X+	Match X one or many time.
X{n}	Match X exactly n times.
X{n,}	Match X at least n times.
X{n,m}	Match X exactly n times, but not more than m times.

Example:

Regular expression: `Cla(s{2})`

Matched text: any sequence starting with "Cla" symbols, followed by "s" symbol two times. It will match any text containing the string "Class".

Boundary Matchers

Boundary matchers help to match strings more precisely. Boundary matchers help by matching a particular word, beginning or end of line, or beginning or end of the input.

^	Beginning of the line.
\$	The end of the line.
\b	A word boundary.
\B	A non word boundary.
\A	Beginning of the input.
\Z	End of the input.

Example:

Regular expression: `\bCla(s{2})\b`

Matched text: any sequence starting with "Cla" symbols, followed by "s" symbol two times. It will match any text containing string "Class" as whole word ("Classs" won't be matched).

Embedded Flag Expressions

Allows setting to set properties for a regular expression matcher.

(?i)	Case insensitive matching.
(?x)	Ignores white spaces in regular expression.
(?m)	Enables multi line option. If not specified, boundary matches ^ and \$ matches beginning of the input and end respectively.
(?s)	Enables expression "." to match any character including line terminators. If not specified, dot in expression does not match line terminators.
(?u)	Enable Unicode-aware case folding. When this flag is specified, case insensitivity is applied to the Unicode standard.
(?d)	Enable Unix lines mode. Only terminator "\n" is recognized in behavior of ".,", "\n", "\$"

Example:

Regular expression: `(?m)^\bCla(s{2})\b`

Matched text: any sequence from a new line, starting with "Cla" symbols, followed by "s" symbol two times.

References

<http://java.sun.com/docs/books/tutorial/extra/regex/index.html>

Replacing

The Find and Replace functionality allows replacing one specified model value with another value.

You can change the values for the following properties:

- Names
- Documentation
- Tag values
- Text included to Notes
- Text included to Text Boxes
- Expressions.

To replace a value:

1. From the **Edit** menu, select **Find and Replace**. The **Find and Replace** dialog box opens.

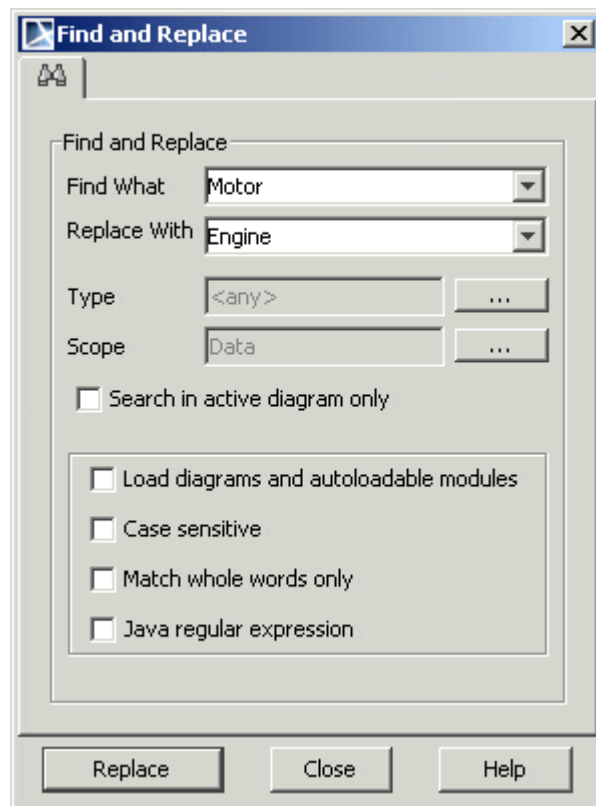


Figure 65 -- Find and Replace dialog box

2. Type the value to be replaced into the **Find What** field.
3. In the **Replace With** field, type the value that will replace the value of the found element.
4. Specify the search criteria. For more information about the search criteria, see "Searching" on page 189.

5. Click the **Replace** button to start the replacement. On each property replacement the question appears. You may choose to replace value, replace all values or not replace.

NOTE

You will see the error when changing value to not valid:

- For example, if value type is *boolean*, you may replace values from *true* to *false* or from *false* to *true*, but if you will try to replace the value *true* or *false* to other, for example, to *Motor*, an error message will be displayed.
- You will not be allowed to change the *Integer* value to *String* when it is a part of the value. For example, if you have the *120* value and trying to replace *20* with *AB*, an error message will be displayed.

Project Partitioning

[View Online Demo](#)[Shared Packages](#)

Partitioning the model

If you developed, or are developing, a large model that has several weakly dependent parts, it is advisable to split it into several module files. Partitioning opens up possibilities for reusing model parts in several related projects and may improve performance on very large projects, when modules are loaded selectively.

Partitioning has a package level granularity. Smaller elements cannot be split into separate modules. In principle each package in a containment tree could be partitioned into a separate module, however this is excessive.

The decision on how to split a model into parts should be made carefully. You should isolate model parts, which form some cohesive, logically complete piece of structure (subsystem, code library, profile) and have light interdependencies.


When there are many one-way dependencies to some model part (parts **A**, **B**, **C** depend on part **D**, but part **D** does not depend on any of the parts **A**, **B**, **C**), this part is a good candidate for placement into module.

4 WORKING WITH PROJECTS

Project Partitioning

When one big project is used to store all the modeling information of the project models (use case models, high level architectural models of the project, detailed implementation level class, sequence, state, etc.), it may be useful to partition the models according to the modeling domains (use cases in one module, architectural models in another, implementation level models in yet another). This allows unloading unnecessary modules while working on one part or another (saving computer and improving performance), but still retain the relationships between domains and load modules, on demand.

NOTE Avoid partitioning a model into parts, which have circular dependencies.

(A  B or A->B->C->A situations)!

Usually programmers are very adept at splitting large code bases into libraries. The very same criteria should be applied to splitting the large models into modules.

MagicDraw module functionality allows two important possibilities:

- possibility to work without all modules loaded;
- read-write modules.

Modules are often used for profile storing, however a module **is not** a profile and it is important not to mix the two. Any model part can be stored in the module.

Exporting the module of a project

NOTE: This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

Using the **Export Module** dialog box, you can partition the model and save the content of a selected package as a separate module. Once exported, the package and its containing elements are read-only and the module name is displayed in brackets next to the package name in the Browser tree.

4 WORKING WITH PROJECTS

Project Partitioning

To export any module using the **File** menu

1. From the **File** main menu, select **Export**, and then select **Module**. The **Export Profile/Module** dialog box opens.

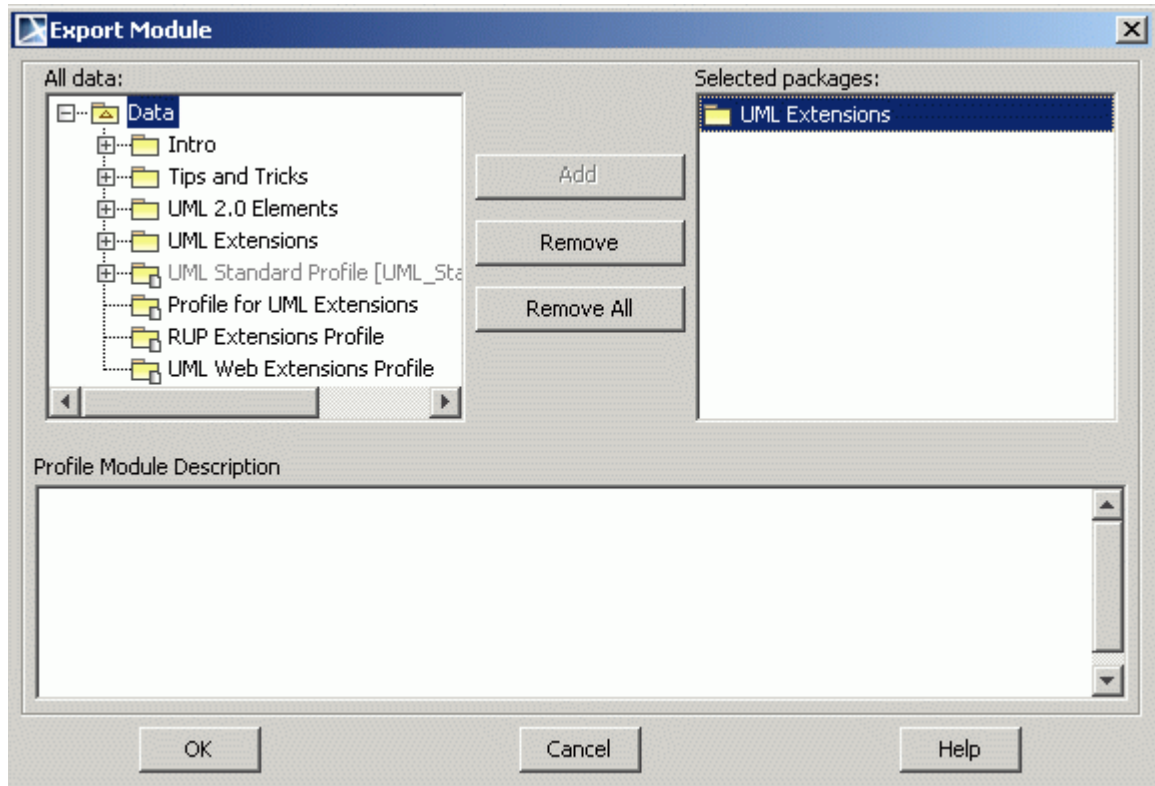


Figure 66 -- The Select Package dialog box for module exportation

2. In the **All Data** list, select the package you want to save as a separate module. Click **Add**. The package is added to the **Selected Objects** list.
3. If desired, type a description of the module in the **Profile Module Description** window. This description is displayed in the documentation of the package.
4. Click **OK**.

To export the selected module using the package shortcut menu

1. In the Browser tree, select the package you want to save as a separate module (you can also select multiple packages).

2. From the package shortcut menu, select **Modules**, and then select **Export Module**. The **Export Module** dialog box opens.
3. In the **All Data** list, select the package you want to save as a separate module. Click **Add**. The package is added to the **Selected Objects** list.
4. If desired, type the description of the module in the **Profile Module Description** window. This description is displayed in the documentation of the package.
5. Click **OK**.

MagicDraw will check for dependencies from the exported part of the model to the part of the model not. You will have to resolve them. The dependency resolution process is the same as for dependency resolution between shared and parts of the module not shared.

When dependencies are resolved, MagicDraw will ask for the file and export the module.

This action can be thought as consisting of 3 elementary steps:

- Saving model elements into the module file.
- Sharing the entire contents of the module.
- Using the module in the main project.

Alternatively, if you have several small, related projects, you can join them together into a larger, partitioned project to work with all the information from one place. This is achieved by using the **Use Module** command that was previously mentioned.

NOTE: Only packages can be exported as modules. To export the created diagram, you must move it to a package containing elements to export.

Sharing the module of a project

NOTE: This functionality is available in Standard, Professional, Architect and Enterprise editions only.

Not all module contents are visible in the project being used. The Module has a shared part and private part. Only contents of the shared part are visible in the project being used. The concept is similar to the public/private parts of modules in programming languages (e.g. Pascal).

To designate packages of the module as shared

- From the **File** main menu, select **Shared Packages**.

- From the package shortcut menu, select **Modules** and then **Shared Packages**. The **Shared Packages** dialog box opens. Use the **Add** button to select more packages for multiple simultaneous sharing, if needed. Click **OK**.

Only the package selected is shared and everything else is not shared.

When the module with shared package(s) is used in the project, the shared part(s) is mounted into the module of the project. Each shared package can have a different mount point. Modules of profiles are typically mounted directly under the top level **Data** element of the package being used, however this can be changed.

Example:

Shared package "*util*" from the module can be mounted on the "*com::company*" path in the main project - to form the "*com::company::util*" path. The Preferred Path of the Shared Package (can be tuned in the **Shared Packages** dialog box) of the module, serves as a hint for MagicDraw on where to mount the package.

Modules form a recursive data structure - the main project uses one or several modules; these modules in turn can use other modules; those other modules can use yet another set of modules and so on. All model pieces from these modules are gathered and connected into the integral model, which is shown in the model Browser when the main project is opened.

Managing modules (the Modules dialog box)

From the **Options** main menu, select **Modules**.

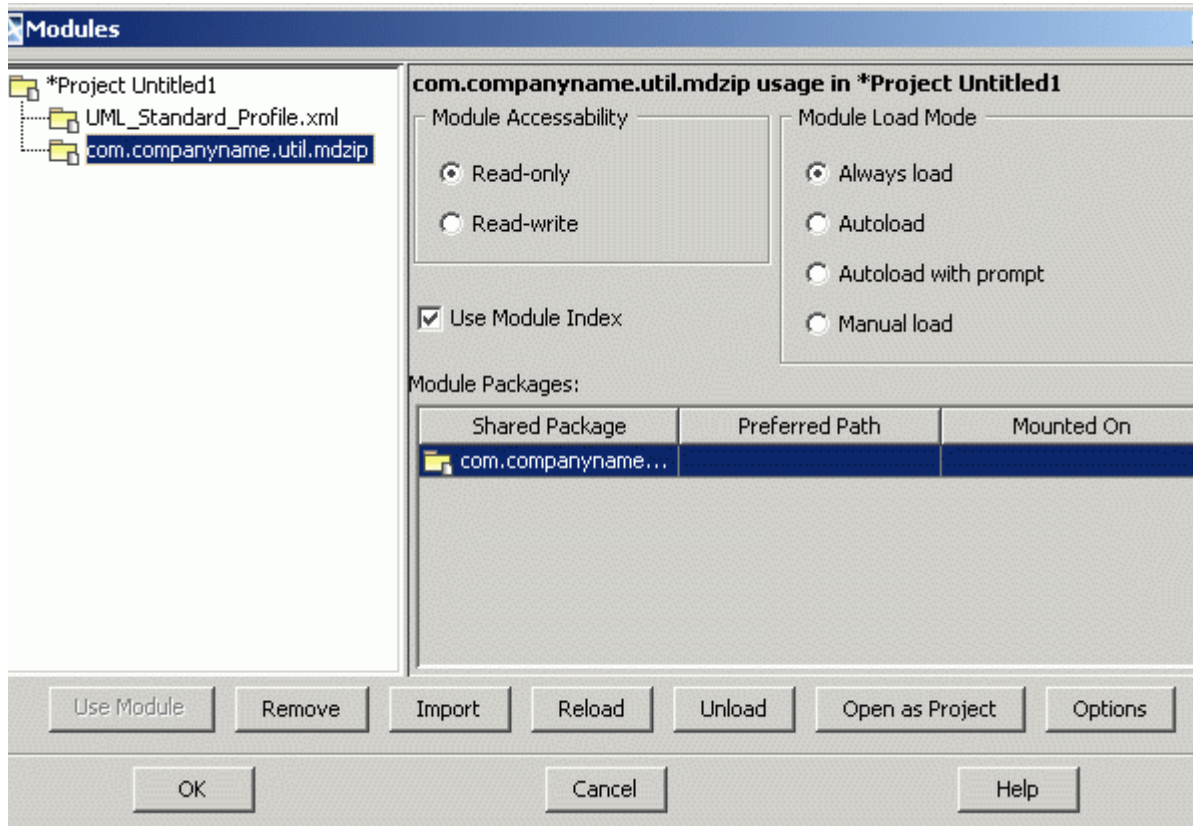


Figure 67 -- The Modules dialog box

Box name	Function
Module Accessibility	<p>Specifies the way a module can be used in a project:</p> <ul style="list-style-type: none"> • read-only modules are not editable within the project using it. • read-write modules can be edited in place - directly in the project using it.

Module Load Mode	<p>Sets the module loading mode:</p> <ul style="list-style-type: none"> • Always load (default) - modules are always loaded when the project is opened. • Autoload - module is not loaded when the project using it is loaded. However, MagicDraw monitors user activities in the project and tries to anticipate guess when the user might want to use the model piece from the unloaded module. • Autoload with prompt - mode is similar to Autoload. However, MagicDraw will ask the user before loading the module. • Manual load - module is not loaded when the project using it is loaded.
Use Module Index	If selected, uses the indexing scope specified in the Project Options dialog. For more information about indexing, see “Indexing” on page 226.
Shared Package	Name of the shared package.
Preferred Path	Carries the information about the path where the shared package should be placed in the project using it.
Mounted On	Holds the packages of the project on which the corresponding module share is mounted. Click the “...” button to change package or create a new one.
Use Module	Enabled, when the module is selected as read-write. Allows the use of the module in the selected module.
Remove	Removes module from the project.
Import	Imports module to the project.
Reload	Reloads module in the project.
Unload	Unloads module from the project.
Open as Project	Opens the selected project as a module.
Options	The Project Options dialog box opens.

Dependencies Between Elements

A package can be exported to an independent module only if it does not depend on external elements (except other modules). Cyclical dependencies between several modules are not allowed.

There are three types of dependencies:

- Dependency by relationship
- Dependency by reference
- Diagram dependencies

Package dependencies by relationship

The module depends on external elements

If a module element has a relationship with an external element and this relationship is contained in the module package, an error message appears when exporting the module.

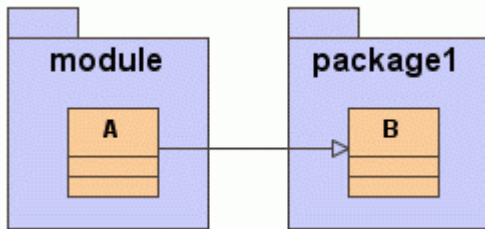


Figure 68 -- Example of a module dependency on an external element

Such dependencies on external elements are displayed in the Browser tree:

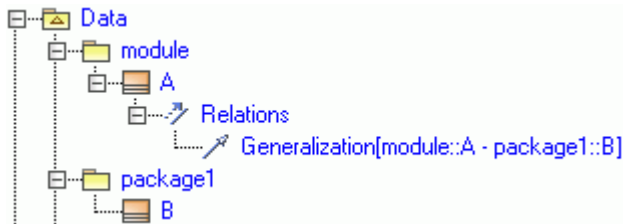


Figure 69 -- Package has a dependency on an external element

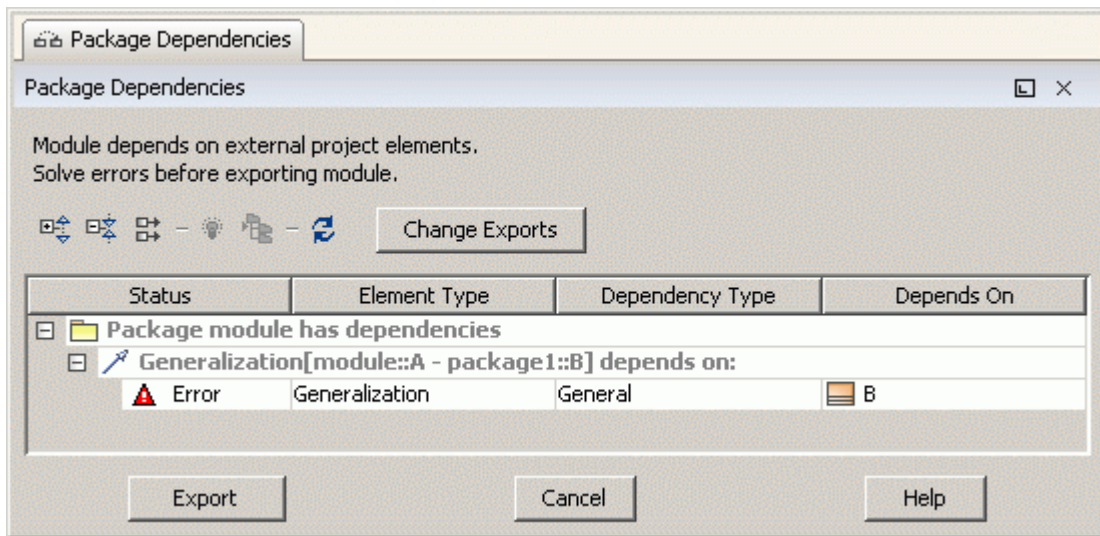


Figure 70 -- Error in the Package Dependencies dialog box

In this situation, MagicDraw can suggest moving the relationship into the parent package of this external element. For example, **package1** is a parent of class **B**, so the relationship can be moved from the **module** into **package1**:

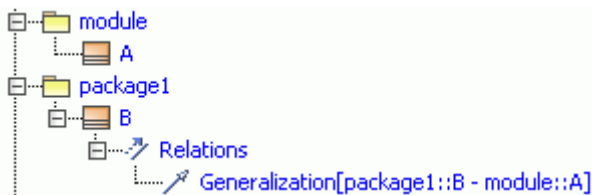


Figure 71 -- Resolved package dependency on an external element

Some movements can be achieved by clicking **Solve** in the **Package Dependencies** dialog box. For a detailed description of this dialog box, see "The module package can now be exported into an independent module." on page 211.

You can also drag-and-drop the relationship from one package to another in the Browser tree.

The module depends on an external element, but can be exported (with warning)

Though the module element has a relationship with an external element, this relationship is contained in an external package:

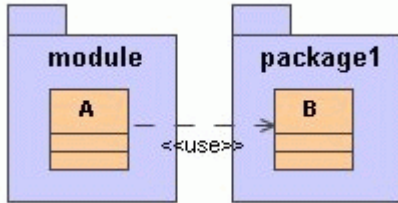


Figure 72 -- Example of a “legal” module dependency on an external element

In this case, the dependency on an external element is displayed in the Browser tree:

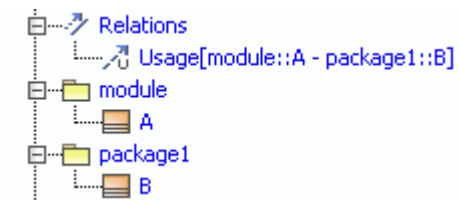


Figure 73 -- “Legal” module dependency on an external element in the data model browser

The package can be exported as a module because the relationship is contained in an external package.

The module does not depend on an external element

If the module element has a relationship with an external element is irrelevant in the context of UML (for instance, the external model uses the module, but not vice versa) and this relationship is contained in an external model, the package can be exported into an independent module:

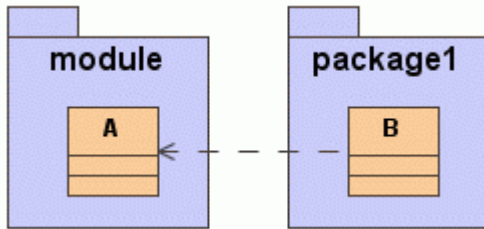


Figure 74 -- Example of a relationship when the module does not depend on an external element

Dependencies by reference

The module depends on external elements when the model elements from the module packages have references to external elements.

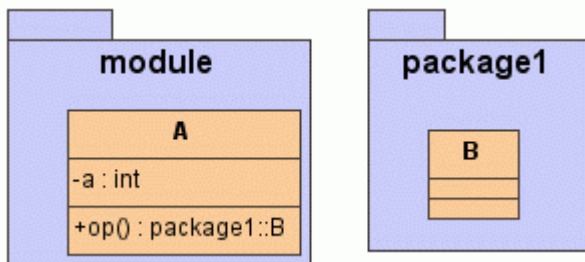


Figure 75 -- Example of a dependency by reference

In this case, the **module** package cannot be exported to an independent module.

Diagram dependencies

The diagram depends on all model elements displayed within it.

4 WORKING WITH PROJECTS

Project Partitioning

If the diagram is contained in a **module** package and depends on external elements, this package cannot be exported to a module.

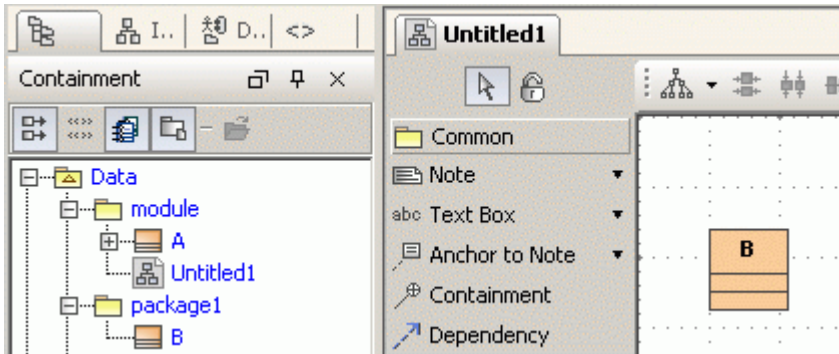


Figure 76 -- Example of the relationship when a diagram depends on an external element

For more information about the package dependencies on external elements, see "The module depends on external elements" on page 207.

In this case, if the diagram is not important to the module, it can be moved from the **module** package into any external package by dragging and dropping it within the Browser tree:

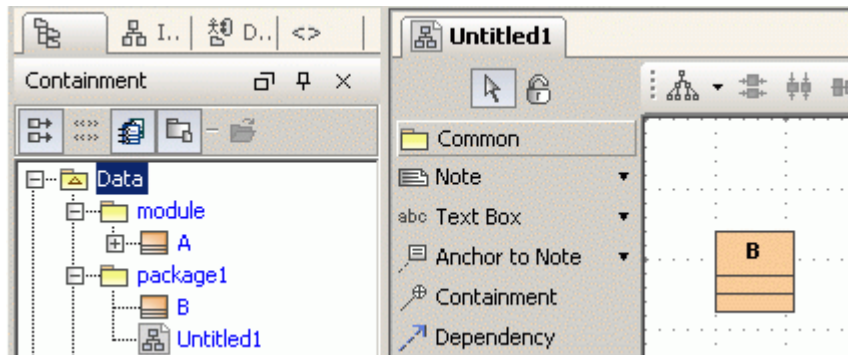


Figure 77 -- Diagram is moved from the module package to the package1 package.

The **module** package can now be exported into an independent module.

MagicDraw Teamwork Server is an ideal solution for group work on the same project. For more information about the Teamwork Server, see the [MagicDraw Teamwork System User's Guide](#).

The Package Dependencies window

To open the **Package Dependencies** window

- Choose the **Dependency Checker** menu item from the **Analyze** menu for dependency analysis of the whole project. This will open a dialog window for configuring dependency checker options (see Figure 78 on page 213). After pressing **OK**, dependencies among the project and shared packages (shared packages may belong to the same project or may be mounted from other projects) will be analyzed and shown in the opened **Package Dependency** panel (see Figure 81 on page 215).
- Choose **Tools > Dependency Checker** from the package or model shortcut menu in Browser or in diagrams to analyze dependencies between the selected package/model and shared packages. This will open a dialog window for configuring dependency checker options (see Figure 78 on page 213). After pressing **OK**, dependencies between the selected package/model and shared packages will be analyzed and shown in the opened **Package Dependency** panel (see Figure 81 on page 215).
- Choose **Modules > Export Module...** from the package/model shortcut menu (module exporting is detailed in section “Exporting the module of a project” on page 201). A question dialog box will be opened (Figure 79 on page 213) asking for your confirmation to start dependency checking between the exported package/model and the rest of the project (including shared packages that belong to the project and used modules). Select the **Check for cyclic dependencies on modules** checkbox, if you want to discover cyclic dependencies*. Press **Yes** to start dependency analysis. Only dependencies which have **Error** and **Warning** (if the **Check for cyclic dependencies on modules** checkbox is selected) severity levels are displayed (see Figure 82 on page 215).

NOTE:

If there is a chain of dependencies such that:

$A \rightarrow M(1), M(1) \rightarrow M(2), M(2) \rightarrow M(3), \dots, M(X) \rightarrow M(A)$, where:

- A is an element from the module M(A)
- M(1..X) are other modules
- $A \rightarrow M(x)$ is element A dependency on module M(x)
- $M(y) \rightarrow M(x)$ is a dependency of at least one element in module M(y) on module M(x),

then this chain is called a cyclic dependency and every atomic dependency in this chain is considered as part of cyclic dependency.

- Choose **Modules > Share Packages...** from the package/model shortcut menu (package sharing is detailed in section “Sharing the module of a project” on page 203). A question dialog box will be opened (see Figure 79 on page 213) asking for your confirmation to start dependency checking between the shared package/model and non-shared part of the project

4 WORKING WITH PROJECTS

Project Partitioning

together with shared packages that belong used modules. Select the **Check for cyclic dependencies on modules** checkbox, if you want to discover cyclic dependencies. The definition of a cyclic dependency is provided in the previous paragraph. Press **Yes** to start dependency analysis. Only dependencies which have **Error** and **Warning** (if the **Check for cyclic dependencies on modules** checkbox is selected) severity levels are displayed (see Figure 82 on page 215).

NOTE: User decisions made in the **Options** window (see Figure 78 on page 213) are remembered between Dependency Checker launches. The settings for this window are retrieved from the **Dependency Checker** options group in **Project Options** (see Figure 80 on page 214).

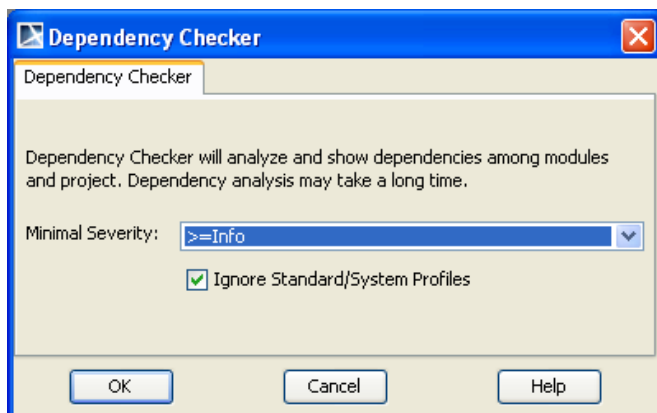


Figure 78 -- The Options window for the dependency checker

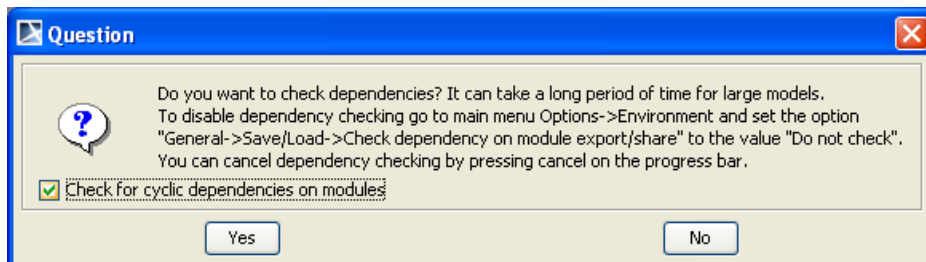


Figure 79 -- A confirmation window for starting up the dependency checker

4 WORKING WITH PROJECTS

Project Partitioning

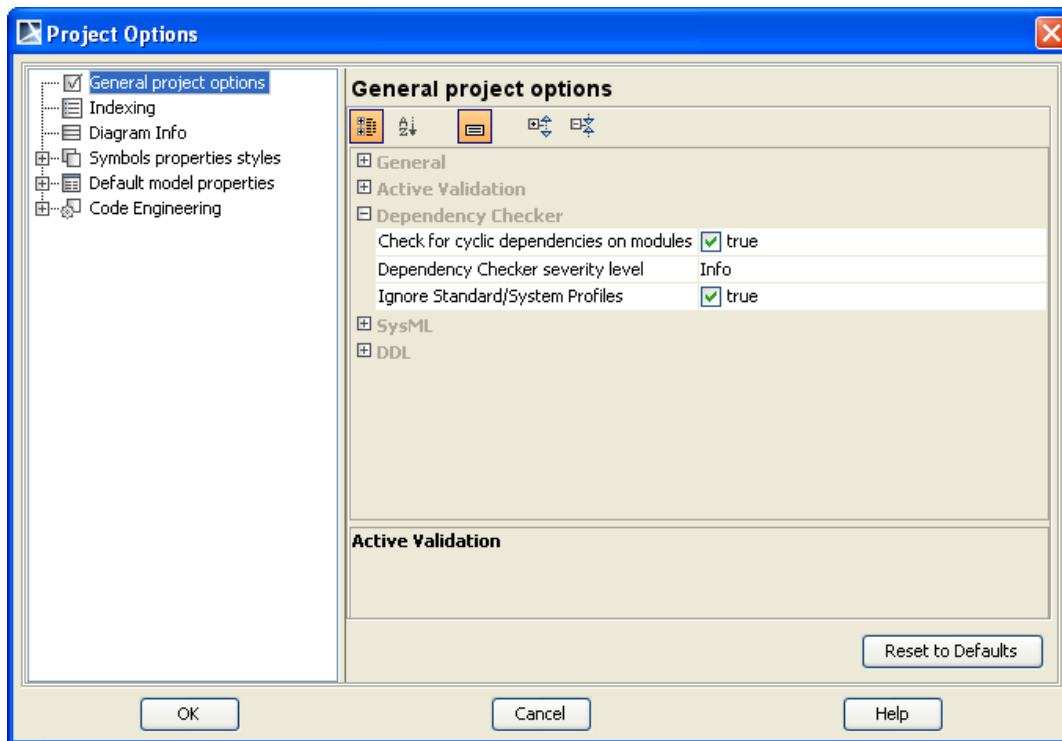


Figure 80 -- The Dependency Checker options group in the Project Options window

4 WORKING WITH PROJECTS

Project Partitioning

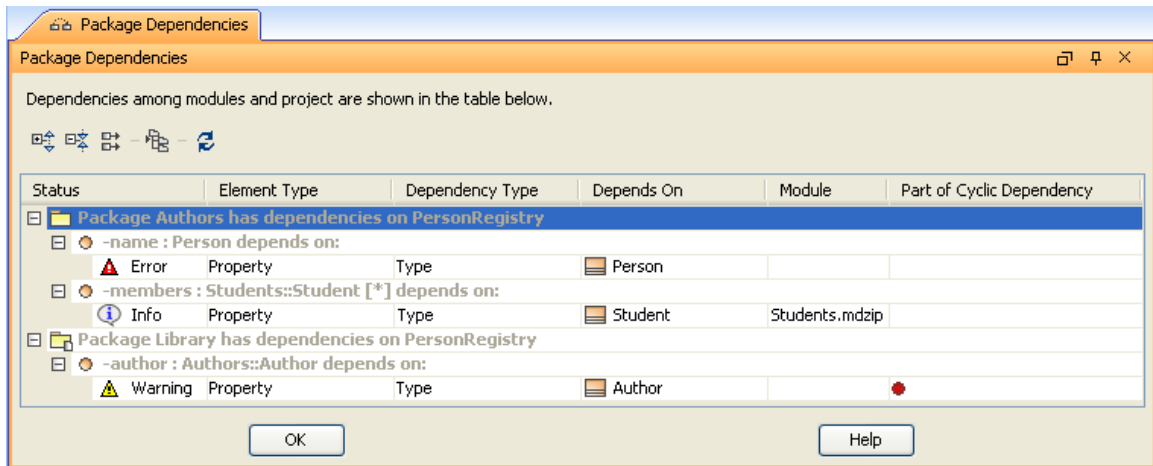


Figure 81 -- The Package Dependencies panel, opened independently

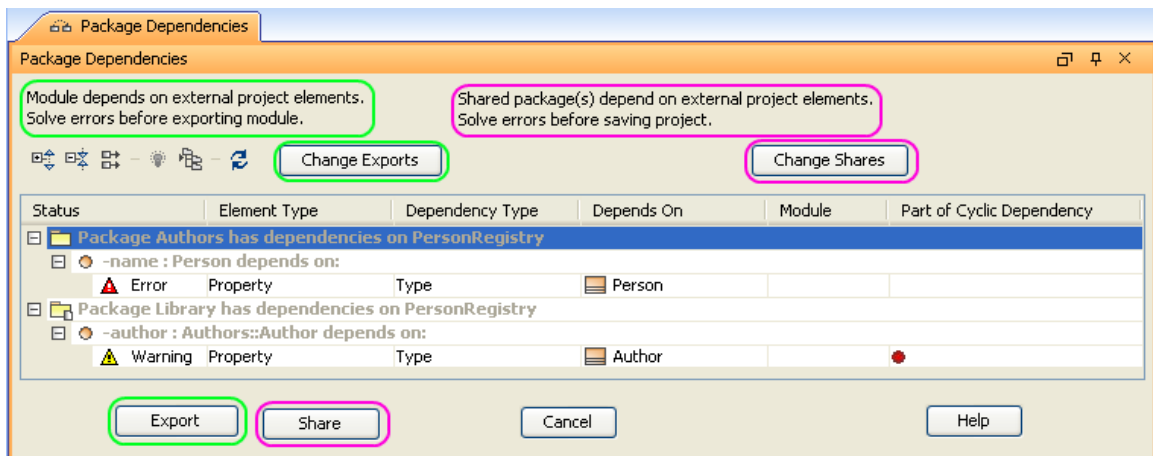


Figure 82 -- The Package Dependencies panel, opened when exporting (GUI elements, highlighted in green) or sharing a package (GUI elements, highlighted in pink)

The **Package Dependencies** panel has a table which shows the list of dependencies and buttons for managing data displayed in this table.

4 WORKING WITH PROJECTS


Project Partitioning

Button	Description
Expand All Tree Branches	Expands all nodes in the package dependencies tree.
Collapse All Tree Branches	Collapses all nodes in the package dependencies tree.
Show/hide the Full Path Names	Displays the element full path next to the element name.
Solve	<p>The button is enabled, when a dependency whose status is Error is selected in the table and a solution for the problem can be found. Clicking the button opens the dialog box for choosing the solution for a specific dependency problem.</p> <p>This button is only visible when exporting or sharing a package.</p>
Select in Containment Tree	Shows the selected element in the Browser. The button is enabled, when a dependency is selected in the table.
Refresh	Performs dependency analysis and refreshes the dependency table with the new analysis results.
Change Shares	<p>Opens the Shared Packages dialog window for reselecting the packages to be shared.</p> <p>This button is visible only when sharing a module.</p>
Change Exports	<p>Opens the Export Module dialog window for reselecting the packages to be exported.</p> <p>This button is visible only when exporting a module.</p>
OK	<p>Closes the Package Dependencies panel.</p> <p>This button is available when the dependency checker is opened independently by selecting Analyze > Dependency Checker from the main menu or Tools > Dependency Checker from the package or model shortcut menu.</p>
Share	<p>Closes the Package Dependencies panel and makes the package shared.</p> <p>The button is enabled, when the environment option Check dependency on module export/share is set to Allow dependencies (Options > Environment from the main menu, General section). Environment options are detailed in section “Setting Environment Options” on page 129.</p> <p>This button is visible only when sharing a module.</p>

4 WORKING WITH PROJECTS

Project Partitioning

Export	<p>Closes the Package Dependencies panel and opens the Save as.../Commit Settings dialog window for saving/committing the package as separate module. The button is enabled, when the environment option Check dependency on module export/share is set to Allow dependencies (Options > Environment from the main menu, General section). Environment options are detailed in section “Setting Environment Options” on page 129.</p> <p>This button is visible only when exporting a module.</p>
Cancel	<p>Cancels package sharing or exporting.</p> <p>This button is visible only when exporting or sharing a module.</p>
Help	<p>Displays MagicDraw Help.</p>

Column	Description
Status	<p>Shows severity of the element dependency problem. The status can be Error, Warning, or Info.</p> <p>Dependencies that have Error status:</p> <ul style="list-style-type: none">• module dependencies on the project <p>Dependencies that have Warning status:</p> <ul style="list-style-type: none">• cyclic module dependencies on other modules <p>Dependencies that have Info status:</p> <ul style="list-style-type: none">• project element dependencies on elements from shared packages (shared packages can belong both to the project and come from an external project)
Element Type	<p>Displays element type.</p>
Dependency Type	<p>Displays dependency type.</p>
Depends On	<p>Displays the model element, on which the package/model element is dependent.</p>
Module	<p>Displays the name of the module file that owns the model element, on which the package/model element depends.</p>
Part of Cycling Dependency	<p>If a symbol “” is depicted, that indicates that the element is a part of the cycling dependency.</p>

Unresolved dependencies

When a model part is exported to a separate module, if there are dependencies from the module back to the project, you are asked to resolve them (dependencies in the opposite direction - elements in project depending on elements in module - are OK).

The same situation occurs when you edit the module inside the project (when the module is mounted read-write on the project) and introduce dependencies from the module back to the project. In this case, you will be asked to resolve these dependencies on module save.

However, it might be inconvenient to resolve these dependencies at that moment (perhaps you have finished work for today and you will resolve dependencies tomorrow, and now you just want to save the project and leave; perhaps the particular dependency resolution is not a trivial task, which will take some time).

MagicDraw allows you to continue without resolving these dependencies. The elements, which were referenced, but are missing in the module will be shown as missing proxy elements (see “Missing elements for the proxies (orphan proxies)” on page 228).

This is one more improvement - in previous versions MagicDraw was strict in checking dependencies and did not allow dangling references. Now more flexibility is allowed.

This behavior is controlled by the **Check dependency on module export/share** environment option (from the **Options** menu, select **Environment, General** section).

There are three choices:

- **Do not allow dependencies** setting restores previous, strict checking.
- **Allow dependencies** is the default setting, described above.
- **Do not check** setting is an even more lax setting; it does not prompt the user to resolve dependencies at all. If you are not careful, this can lead to the proliferation of missing proxy elements, hence proceed with care.

Using the module of a project

When a module is used in another project, its contents are linked-in and made accessible in the model tree of the project using them as if it were part of the project.

To use a module in a project

1. From the **File** main menu, select **Use Module**.
2. In the **Use Module** dialog box, select the module you want to use in your project, specify the module settings and click **OK**.

The model elements are still stored separately; module elements - in the module file and main project elements - in the main project file.

The Use Module Wizard

From the **File** menu, select the **Use Module** command.

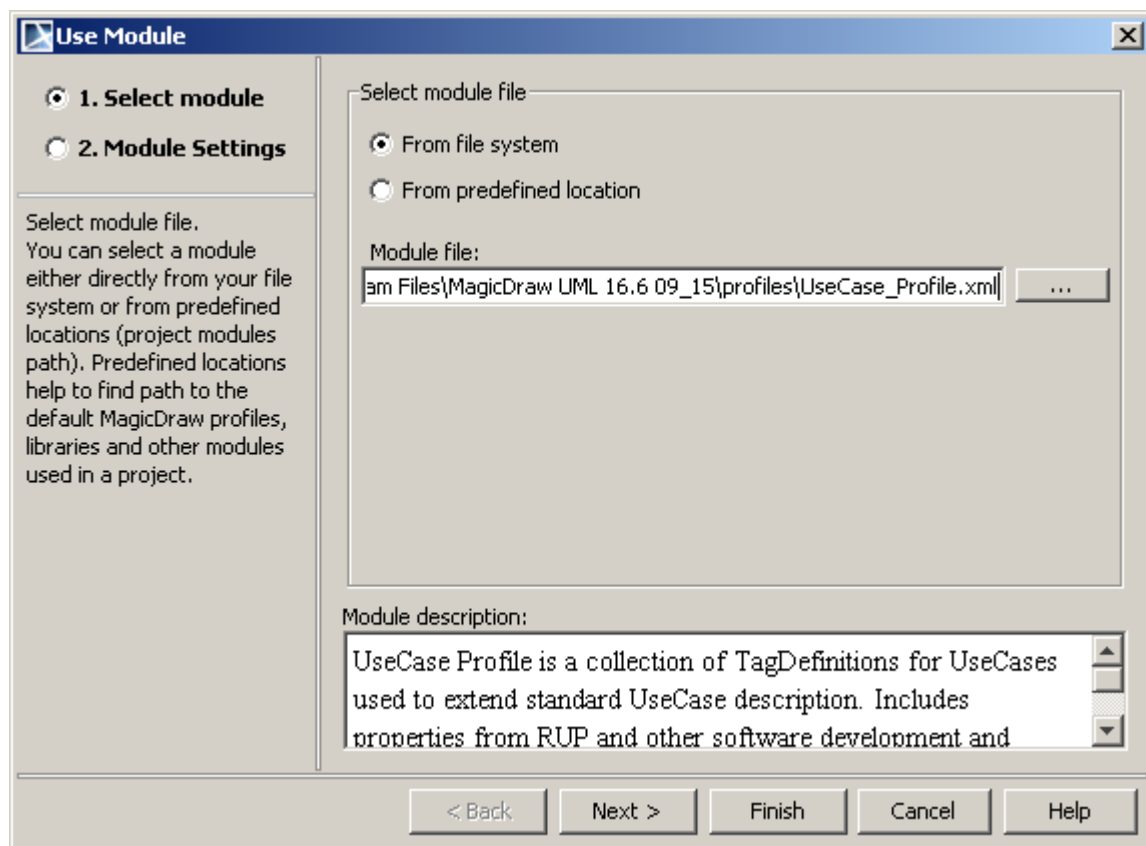


Figure 83 -- The Use Module Wizard, Select Module step, the From file system radion button

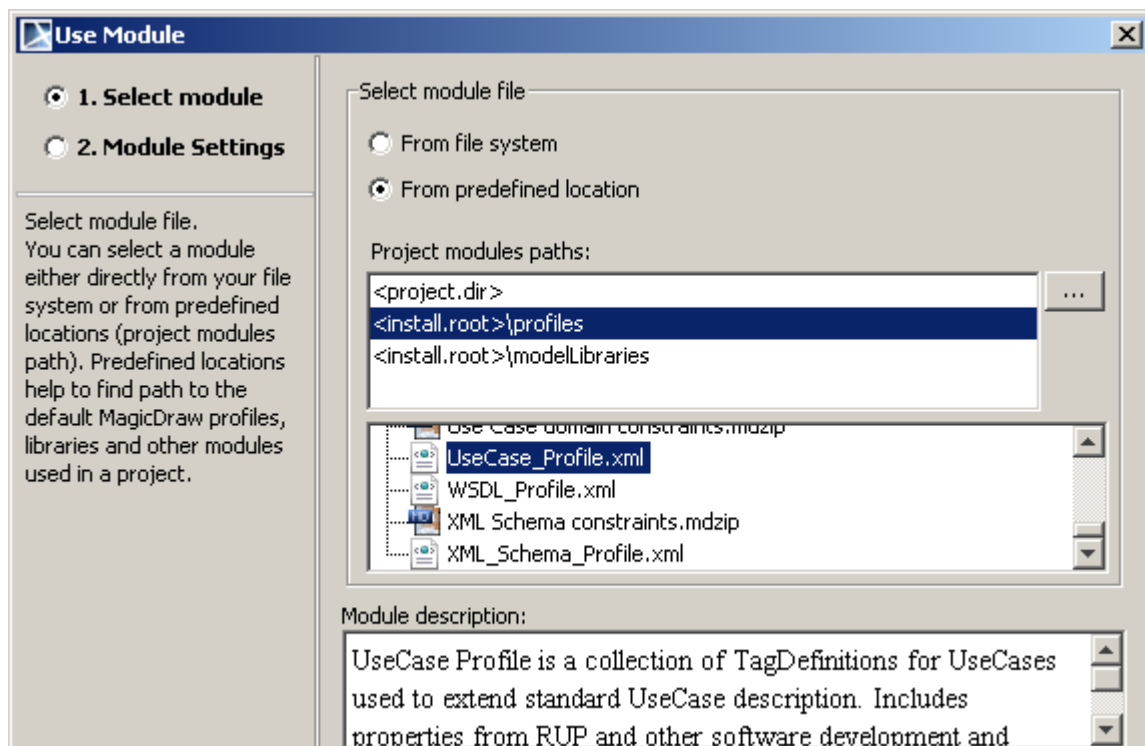



Figure 84 -- The Use Module Wizard, Select Module step, the From predefined location radion button

Box name	Function
 From file system	Allows selecting module file from your file system. Click the "..." button next to the Module file text box, to browse to module file (Figure 83 on page 220).
From predefined location	Allows selecting modules from predefined locations. Select project module path from a paths list, and then select module file from the list below. Click the "..." button next to the Project modules paths text box to add new module path to your project. The Select Folder dialog box will open (Figure 84 on page 221).
Module description	Displays the module description.
Next	Proceeds to the next step.

4 WORKING WITH PROJECTS

Project Partitioning

Finish	Saves changes and closes the dialog box.
Cancel	Cancels the dialog box without saving changes
Help	Displays MagicDraw Help.

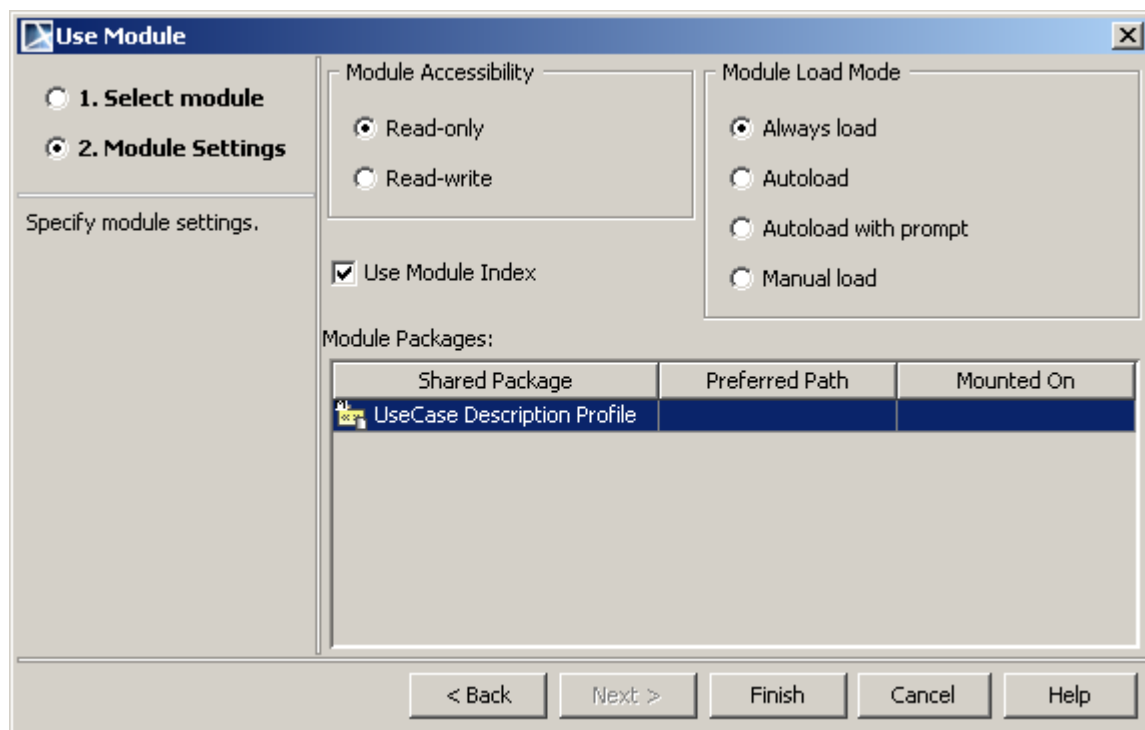


Figure 85 -- The Use Module Wizard, Step2 - Module Settings

For more information about the options in this step, see “Managing modules (the Modules dialog box)” on page 205.

Reusing model parts between models

When models are properly partitioned, model parts can be reused in other projects.

4 WORKING WITH PROJECTS

Project Partitioning

For example, a common situation in Java software projects is this layout of the packages (in project **A**):



Common and *util* packages are good candidates for refactoring into modules. Then in project **B** these modules can be reused.

There are two ways to use the module in the project:

- read-only modules are not editable within the project using it.
- read-write modules can be edited in place - directly in the project using it.

The usage mode can be specified in the **Use Module** Wizard, Step 2. By default, the module is used in the project in read-only mode.

To change module accessibility mode

1. From the **Options** main menu, select **Module**. The **Module Options** dialog box opens.
2. Select module in the tree and change the **Module Accessibility** option from read-only to read-write or vice versa. Click **OK**.

You can change content of a module and make its inner elements editable by selecting **Open Module As Project** (from the module shortcut menu, **Modules** submenu). The module opens as a separate project.

When to use read-only module?

The decision to use a module as read-only or read-write depends on the maturity of the module and the organization ownership/responsibility rules for the projects, developing modules.

If the library in the module is mature (changes to it are not expected/likely/possible) it should be used in read-only mode.

If the module is owned by a team, working on one project, and this team is responsible for this module and the module is reused in another project, the module should be used in the other project as read-only. This prevents inadvertent changes to the library.

When to use read-write module?

In the case where a module is actively developed and evolves together with the projects that are using it, a module should be used read-write.

In this case, if there are multiple projects using the module, you should be careful and remember, that your changes to the module will be reflected in other projects, therefore care should be exercised. Usage of teamwork server might be advisable in this case. And, of course, there can be mixed usage situations - when a module is used read-only in some projects and read-write in others.

Reloading the module of a project

The best way to access the latest changes to your module is to reload it. All modifications made in the other project for this module and then exported as modules with the same name, are reloaded in the current project.

To reload the module

In the Browser tree, from the exported module shortcut menu, select **Modules**, and then select **Reload Module**.

NOTE: If you open your module as a project, be sure to save any changes you have made (by using the Save command). All modifications appear after reloading the module in the other open project, which includes your module as a component.

Importing the module of a project

If you wish to store model elements of the module in the main project file, you can import the module into the project.

To import a module into a project

From the package shortcut menu in the Browser, select **Modules** and then **Import Module**.

All the model elements from the module will be copied into the main project, and the module will be unlinked from the project.

Working with partially loaded projects

In MagicDraw v11.5, the possibility to work with some modules unloaded was added. Prior to this release, all the modules were required to be loaded when the main project was loaded. If the module file was missing, it was an error.

This feature allows some memory to be saved and improved performance when working with very large projects. "Large" for MagicDraw is several thousands of classes and other complex elements. If counting all the small elements, such as properties, methods, method parameters, ~50K of elements is considered a large project. A good example of a large module is a module having a model of Java rt.jar reversed into it. Also diagrams are large elements - 20 or more complex diagrams should be considered large.

When working with a large project partitioned into several modules, at any moment a module can be unloaded. When editing a project, if you see that the module will not be used for some time (perhaps you are working on a different part of the large project), you can unload it - this will save resources.

To unload a module from a project

From the module shortcut menu, select **Modules** and then **Unload Module**.

An unloaded module can be loaded at any time.

To load a module in a project

From the module shortcut menu, select **Modules** and then **Load Module**.

When the module is unloaded, there are some model elements left in the place where the module was mounted. These elements are not editable, and they have a small M in the upper right corner of their icon.

These are the so-called "proxy" elements of the real elements from the module. Instead of the real model elements, the proxy carries only the name and kind of the model element information - it is a lightweight surrogate for the real model element. The proxies are left in the place of those module elements, which are referenced from the main project. These proxies are normal and necessary to maintain project integrity (so that there are no dangling ends of relationships, types of properties do not disappear, etc.).

There are 4 module loading modes:

- Always load (default) - this mode of operation closely mimics the pre-v11.5 MagicDraw functionality. In this mode, modules are always loaded when the project is opened. They can be unloaded if the user deems it necessary.
- Autoload - module is not loaded when the using project using it is loaded. However, MagicDraw monitors user activities in the project and tries to anticipates when the user might want to use the model piece from the unloaded module. E.g. if the user does the search, finds usages/dependencies, reports, metrics, transformations, or code engineering actions with a scope that touches the unloaded module, MagicDraw will load the module.
- Autoload with prompt - mode differs from the *Autoload* mode in this way: MagicDraw will ask the user before loading the module.
- Manual load - module is not loaded when the project using it is loaded. It can be loaded, using the aforementioned **Load Module** command.

To change the module loading mode

1. From the **Options** main menu, select **Module**. The **Module Options** dialog box opens.
2. Select a module in the tree and change **Module Load Mode** by selecting the appropriate radio button.

Modules, which are used very frequently, should be set in the *Always load* mode.

Modules, which are used only occasionally, should be set in the *Autoload* mode (or *Autoload with prompt* if you like to have more control on the loading behavior).

Modules, which are used only very rarely, can be put in the *Manual load* mode. Another frequent case where modules can be set into *Manual load* mode is when modules represent some software library, which is not expected to change. See the paragraph 1.6.1 Indexing below.

Advanced Concepts

Indexing

Indexing can be considered as an intermediate form of work, between working with a fully loaded module and working with the module unloaded.

When a module is unloaded/not loaded in the project, only necessary proxy elements are shown in the place of the module. However, there is a possibility to retain more proxies from the unloaded module than is kept by default. There is one case, where this functionality is particularly useful.

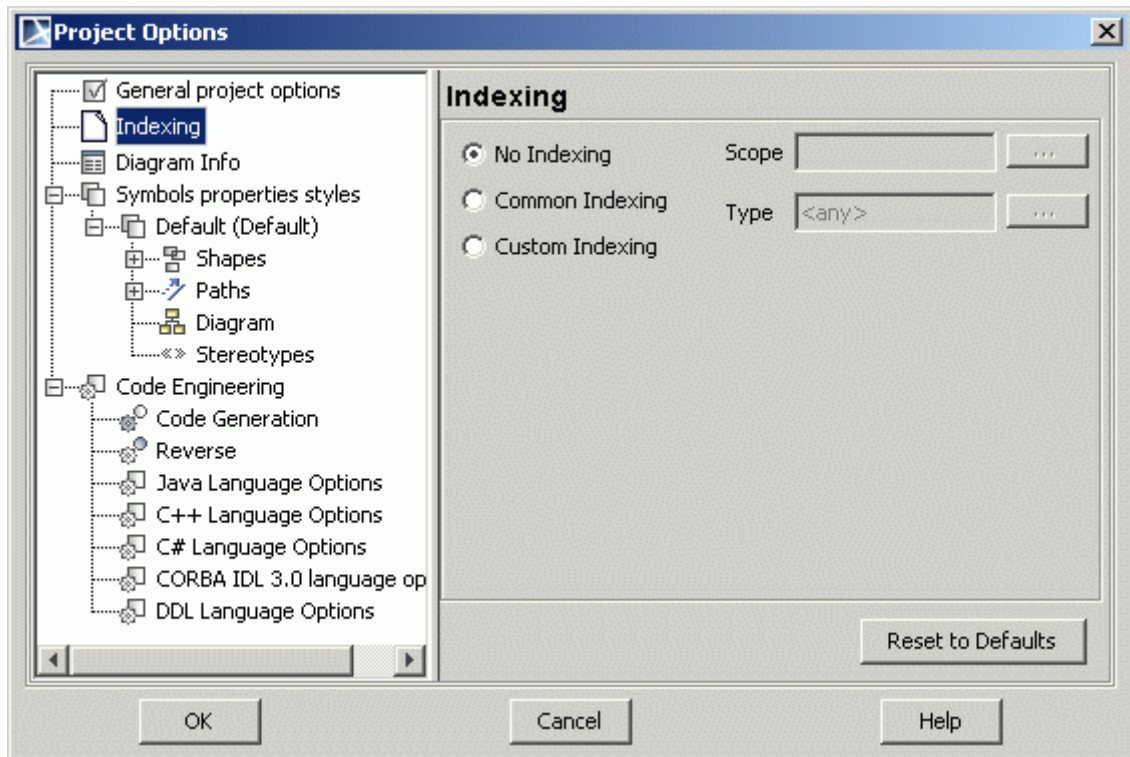
4 WORKING WITH PROJECTS

Project Partitioning

Consider the large software library module in a project. Let's say, only the various classes are used in the main project - some library classes are set as types of properties in the model classes, some model classes inherit from the library classes, etc. In this case, structural information of the library classes (their properties and methods) is not important. If proxies of all classes could be retained when the module is unloaded, this library module could be used in the main project in the unloaded state (saving a considerable amount of computer resources). The indexing feature allows achieving this functionality.

To specify indexing scope

1. First open the module as a project.
2. From the **Options** menu, select **Project**. The **Project Options** dialog box opens. Select the **Indexing** section.



3. Select the **Common Indexing** radio button. This enables indexing of the module and determines what information will be indexed.
4. When common indexing is chosen, classifiers and their inheritance relationships will be indexed. If you want more elements to be indexed, select the **Custom Indexing** option and

fine-tune what element types (properties, methods, etc) should be indexed. The more elements you select, the more elements will be accessible in the project using them as proxies. However, your gains in performance from the module unloading will also diminish. Hence, a balance is needed when customizing the index. It is usually better to use the common indexing variant. Click **OK**.

5. In the project using the indexed module, go to the options of this module (**Options** menu - **>Modules**) and select the **Use Module Index** checkbox for that module.

Such setup causes all the classes of the module to be visible as proxies when the module is not loaded (it is also advisable to change the loading mode of this module to Manual loading).

These proxies can be used as normal model elements in the project using them, without ever loading the module. They can be set as types of properties of the classes in the main project, they can be set as an association ends, classes may be derived from them, etc. If you ever need more information from that module, you can load it at any time to access the full data in the module.

An example could be in the module holding standard Java classes (rt.jar was reversed into it). This module is large, having all the details of standard Java classes. Many of these details are unused in the project; frequently only class information is used in the project for modeling tasks.

Missing elements for the proxies (orphan proxies)

Orphaned proxy (indicated with [!] adornment in the model tree) is really an indication of the dangling reference. Appearance of the proxy is indication that some other elements for example from outside the module (i.e. elements in the main project or other modules) reference to the element in the module that was previously there but no longer exists. Element was deleted/removed/somehow made unavailable in the module. In such case MagicDraw creates so called "orphan proxy" in place of

the missing element - a surrogate/not real element (with exclamation mark - "!") in place where real element was once in the past.

Searching for orphan proxies

Run a search (**Ctrl+F**), check the **Orphaned Proxies Only** checkbox. This will give you all the orphan proxies in your model in the search results.

Resolving orphan proxies

To resolve the orphan:

- Right-click the orphan in the search result or containment tree>Orphaned proxy Resolution.

There are 3 actions you can do with the orphan proxies:

1. Clear Proxy Usages. Can also be triggered by simply pressing the **Delete** button. This action clears all the references to this non existing element, hence there is no more need for the proxy to appear.
2. Replace With. This replaces all the references to the orphan with references to a chosen element.
3. Create New Substitute. This "resurrects" the element which is missing (module must be editable for this).

Which one of them to choose depends on wherever these missing elements are necessary or are they unnecessary.

If they are necessary (i.e. you want these elements to exist; they must be there), this means that they were deleted by mistake at some point in the past. Maybe MagicDraw can not find the required module? Or elements were moved to some other module? Or maybe the old version of the module is used, which hasn't got these elements? The causes may be numerous.

You have to find where these elements are in your modules/main project.

- If they were moved to some other module, you have to use that module into your main project (File>Use module).
- If elements were in the part of the module which was unshared, share this module part again.
- If MagicDraw can not find the module on disk, it should ask you to provide path to it on project load.
- If elements were deleted from the module/main project, you have to roll back to the previous version of the module/project (in your version control system or Teamwork server, or wherever you back your files up into) which still had these elements.

Creating New Substitute

As the last resort, if you have no version saved, where these elements still exist, you can try to "resurrect" them. Right-click each proxy>**Orphaned Proxy Resolution>Create New Substitute**.

MagicDraw will recreate the missing element from the bits of information it still has (which is not much - ID, name and kind of the element).

NOTE: This action might be disabled if the substitute to be created must be in the module, but this module read-only. In this case simply change module to read-write in the **Module Options** dialog (**Options>Modules**) and the action will be enabled.

Deleting orphan proxies

If these elements are unnecessary (i.e. you want them to disappear; they must not be there), this means that they were deleted properly. Now all we have to do is clear the dangling references, which still exist in the other modules/main project to these non existing elements.

To delete orphan proxy:

- right-click each proxy>**Orphaned Proxy Resolution>Clear Proxy Usages**;
- or press **Delete** on the selected orphan proxy.

You can also do this en masse:

1. run a search (**Ctrl+F**);
2. check the **Orphaned Proxies Only** checkbox. This will give you all the orphan proxies in your model in the search results;
3. select them all and press **Delete**.

When references to them are cleared, orphan proxies will disappear.

NOTES:

- Note that if references to these non existing elements are in the modules, which are mounted read-only, this action can not clear them.
- Clear Proxy Usages or Del button resolution method works temporarily only- when project is loaded next time orphan proxies will reappear. To delete proxies fully you have to open each module as project and clean orphan proxies there.



Ecore Support

Introduction

Ecore model is one type of models that Eclipse Modeling Framework (EMF) supports. This type of models is also colloquially called EMF models (even though EMF supports many types of models e.g. UML model). Ecore models can be used for various purposes. Ecore models can be used for meta-modeling purposes, where it's expressive power is roughly similar (slightly higher) to EMOF. Ecore can also be used for simple class modeling purposes as a subset of UML.

MagicDraw has Ecore **export** functionality. Ecore models, prepared in MagicDraw can be exported as Ecore models for further processing with other EMF tools - generation of model repositories, code, XML parsing and storing etc.

Preparing Ecore Models

To start modeling in Ecore, you can use "new project from template" functionality. Select **File>New** from main menu, then choose **Project from Template** project type, and select the **Metamodeling>Ecore Template** template.

If you already have some UML model or CMOF/EMOF model, which you want to develop into Ecore model, you can just attach the Ecore profile from MagicDraw's profiles directory (select **File>Use Module** from main menu, choose **Ecore_Profile.xml** from MagicDraw profiles). If your Ecore model references some standard Ecore elements (such as standard data types - like **EShort**, or standard metaclasses - like **EStructuralFeature**), you also need to use standard Ecore library (select **File>Use Module** from main menu, choose **Ecore.mdzip** from MagicDraw model libraries).

There are no separate custom diagrams for Ecore model editing. You can use the same Class diagrams as you use for your UML models. Since Ecore is almost a subset of UML (with a few additions), familiar UML elements are used for modeling. You can also develop Ecore models without using Ecore profile at all - if your Ecore model uses only UML-specific information, you can develop it using plain UML and export it to Ecore without a problem.

Ecore is even more similar to EMOF. Ecore profile is built on and depends on Metamodeling profile (which implements EMOF modeling). Element properties, that are relevant to MOF (Package namespace URI and Property isID settings) are also relevant in Ecore. You can also have one model and export it as Ecore and then export it also as EMOF.

Class, DataType, Enumeration, Package, Operation, Parameter have direct one-to-one correspondence between UML and Ecore.

Ecore has two flavors of structural features - EAttribute and EReference where UML has just one - Property. Fortunately differentiation between attribute and reference is unambiguous and automatically resolved (property, whose type is data type is treated as EAttribute; property, whose type is class is treated as EReference), hence user does not need worry about this - he can simply use properties.

There are no standalone Association, Generalization model elements in Ecore, but there is analogous information in Ecore (two EReferences, pointing to each other by their **opposite** fields is equivalent to association; EClass::**eSuperTypes** field is equivalent to generalization). Hence it is possible and meaningful to draw associations and generalizations in your model for exporting this information to Ecore.

Ecore generics (templates) are also supported. You can use UML template support to model Ecore generics. While modeling is not trivial (and not one-to-one due to weak semantics of Ecore's EGenericType), it is possible to model all cases of template types, even ones with complexly nested type bounds - like for example, **SortedList<T extends Comparable<? super T>>**.

Your models can also contain any other UML elements, which are not present in Ecore. These elements are simply skipped during export to Ecore. A warning is given about these elements - see "Validation" on page 242.

There are a few Ecore-specific properties, which are brought in when Ecore profile is used. These properties are used to capture Ecore specific information, not existing in UML. MOF-specific properties are also relevant for Ecore. These special properties are overviewed below:

Ecore package has an additional **nsUri** (shared with MOF models) and **nsPrefix** fields:

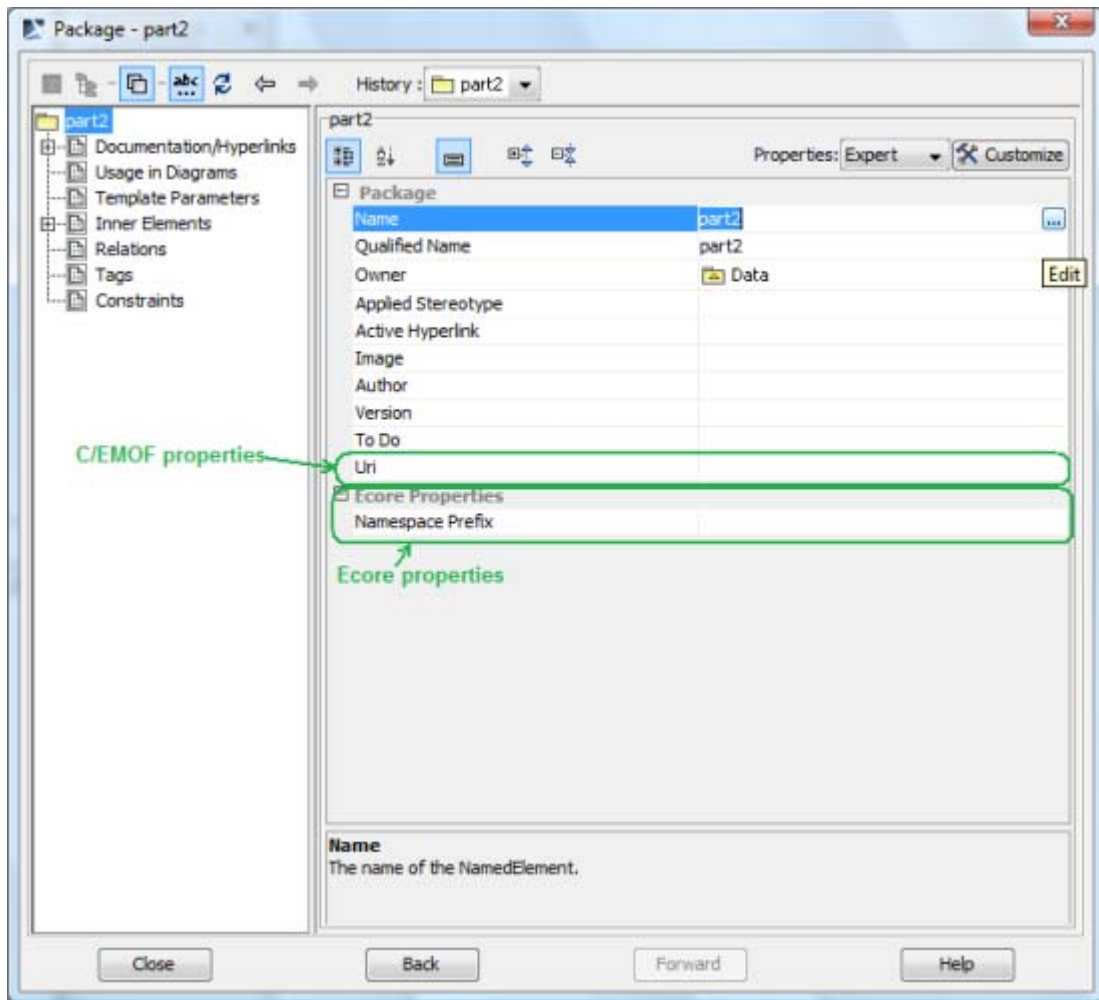


Figure 86 -- Ecore Package Additional Properties

Ecore classifier (class, data type, enumeration) has an additional **instanceClassName**, **instance-TypeName** fields:

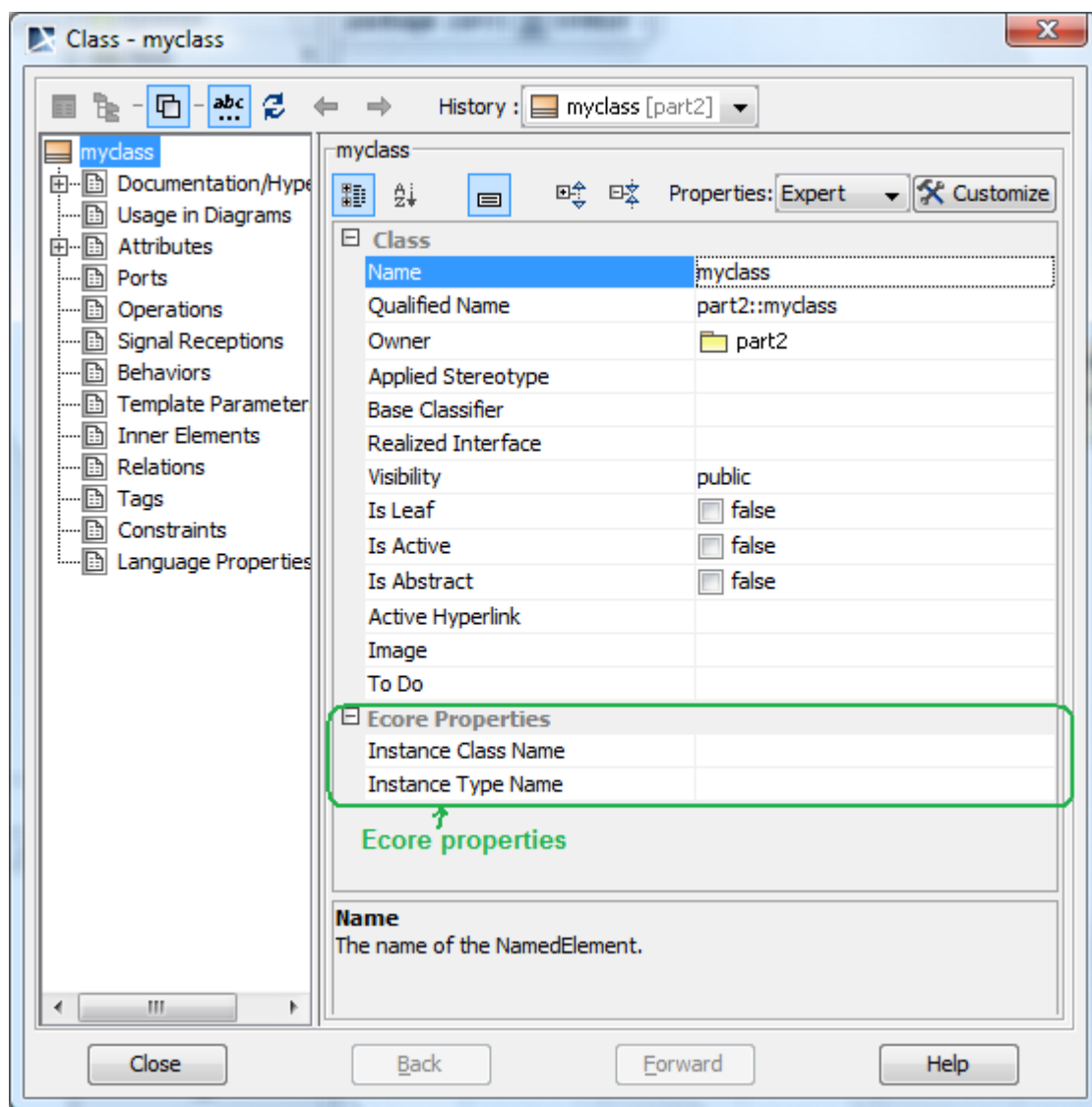


Figure 87 -- Ecore Classifier Additional Properties

Ecore attributes and references (modeled as UML property) have additional **volatile**, **transient**, **unsettable**, **ID** (only for attributes, shared with MOF models), and **resolveProxies** (only for references) fields:

4 WORKING WITH PROJECTS

Ecore Support

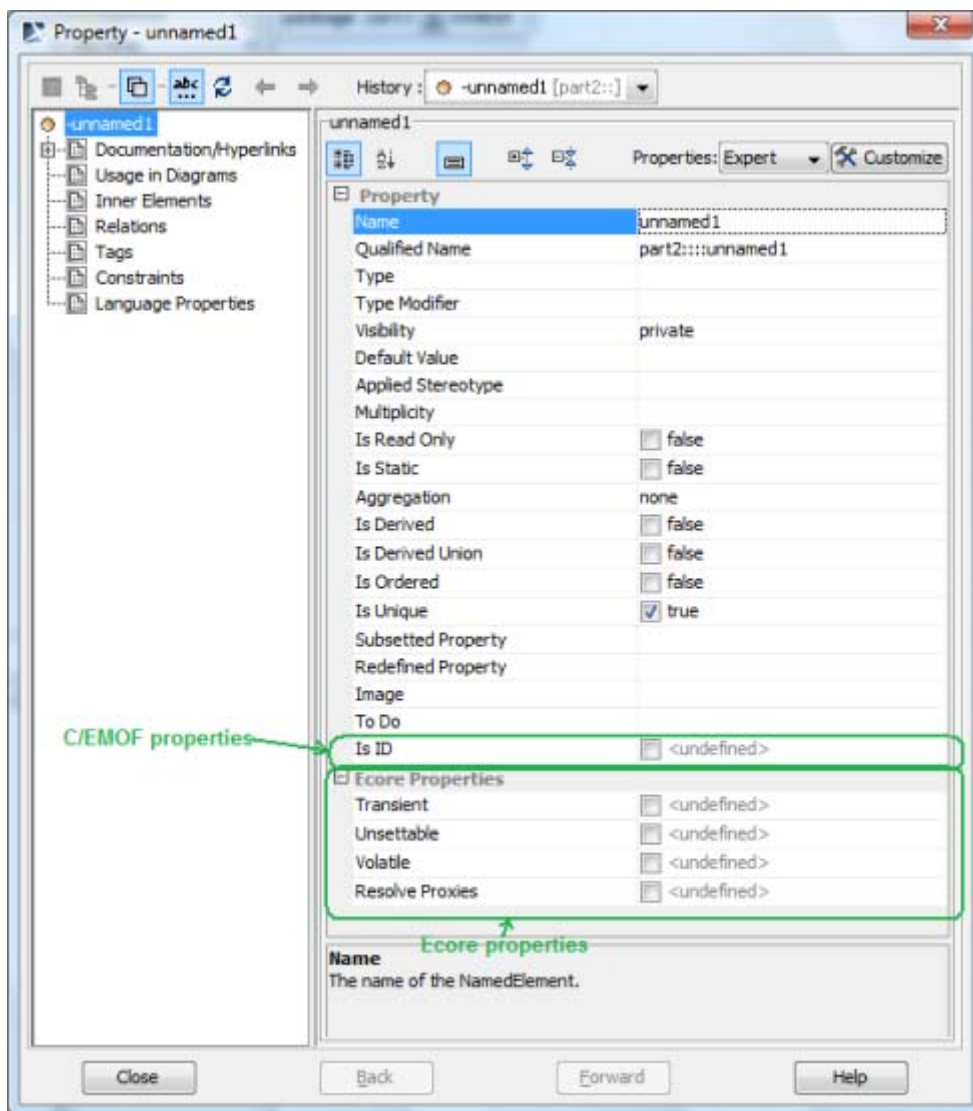


Figure 88 -- Ecore Attribute, Reference Additional Properties

Annotation modeling

Ecore annotations are modeled as UML comments. For simple annotations no additional actions are necessary.

However Ecore annotations have more powerful semantics than UML comments - they can have internal substructure. In particular they can have an additional key-value map. For this additional information, there is a special <<EcoreAnnotation>> stereotype, that can be applied on an annotating comment. After this application, key-value map can be entered in a separate node of the specification window. Key-value pairs are stored as internal subcomment elements of the annotation. This process is illustrated in the pictures below:

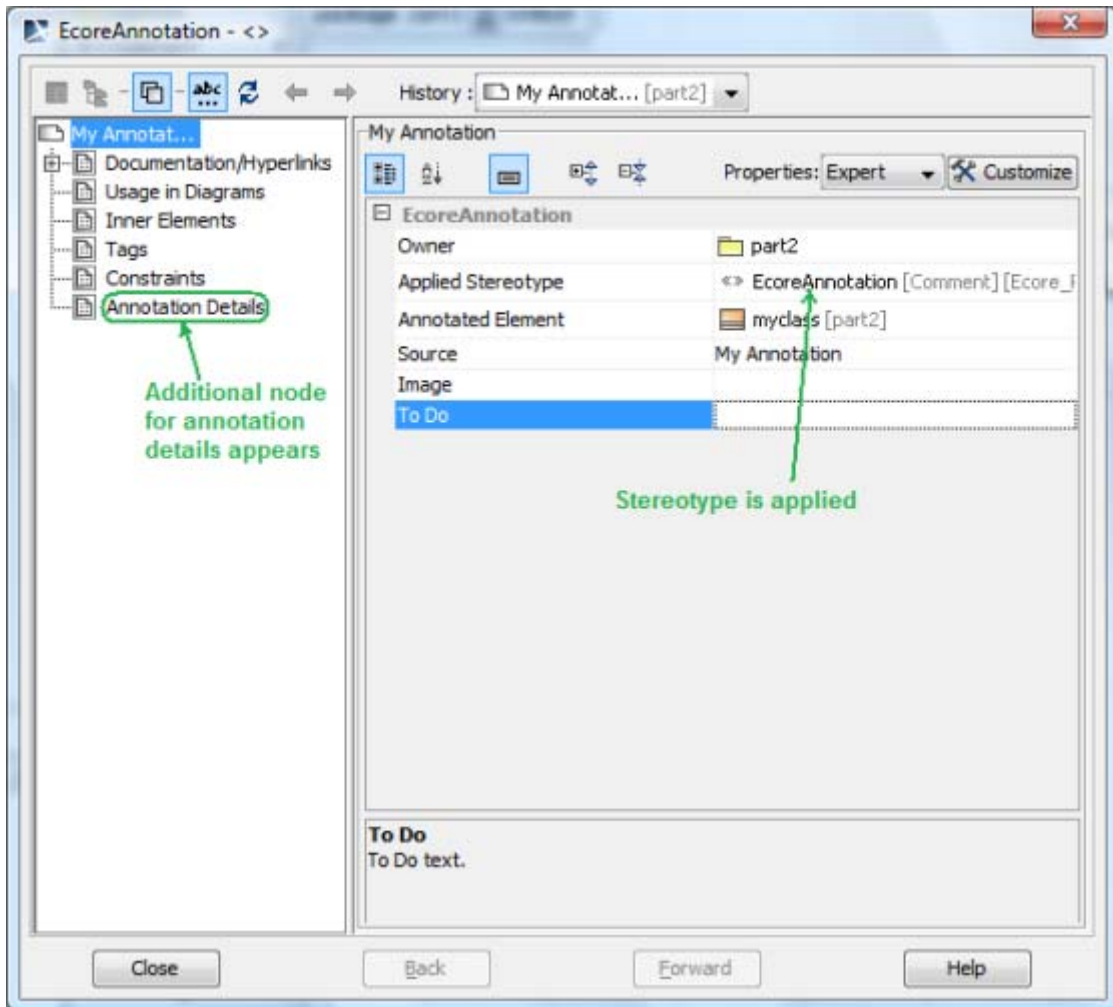


Figure 89 -- Applying EcoreAnnotation Stereotype

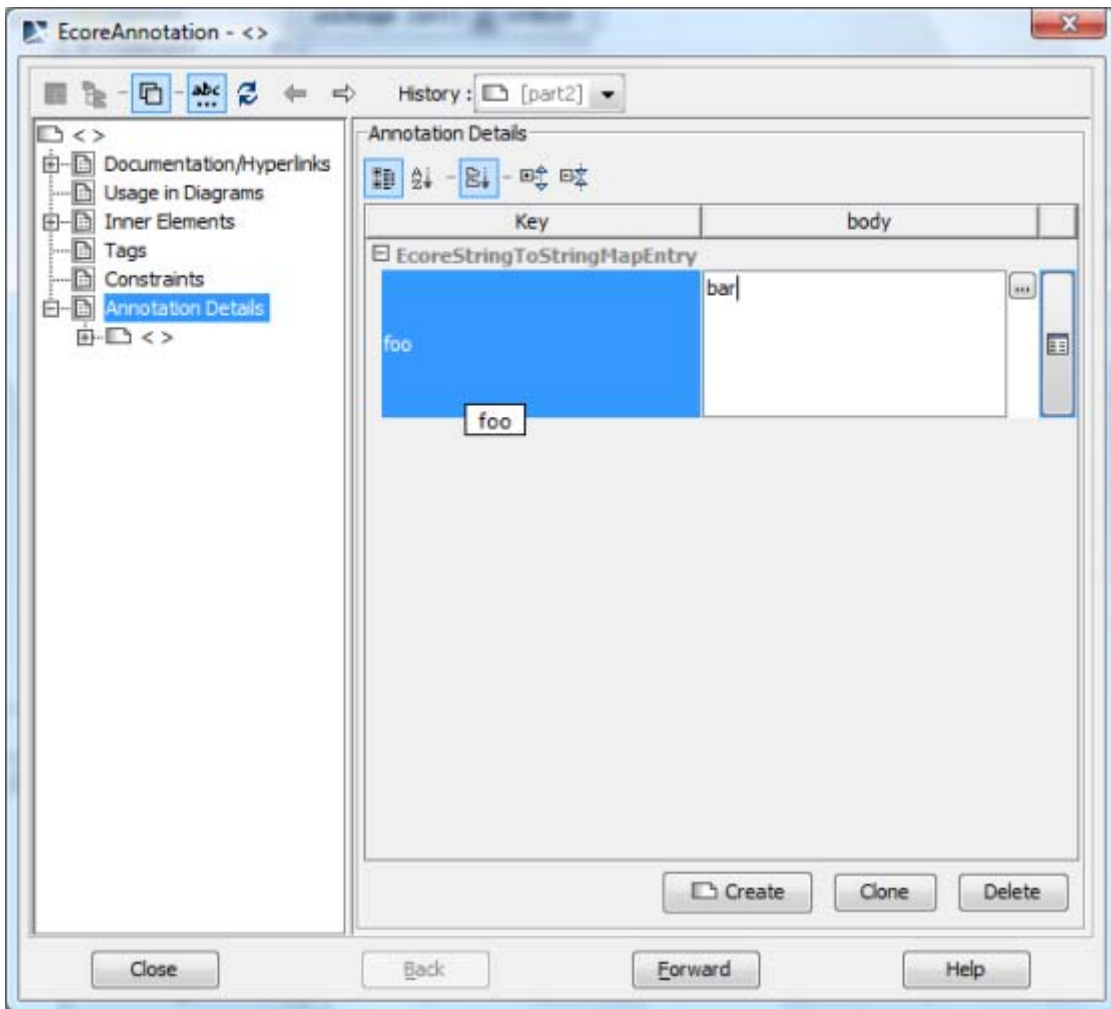


Figure 90 -- Filling In Annotation Details - Key - Value Map

Ecore annotations also have one additional feature - they can also hold an arbitrary content in their **contents** field. This field can hold anything - up to and including entire submodels. **contents** field modeling is not supported in the current MagicDraw release.

Exporting

Exporting of Ecore models is very similar to exporting of the EMOF/CMOF models. After Ecore model is prepared, you can export either whole model or only selected package(s) to *.ecore file format.

To export whole model, use **File>Export To>EMF Ecore File>Ecore Whole Model** menu item. Choose destination file in the opened dialog and press **Export**.

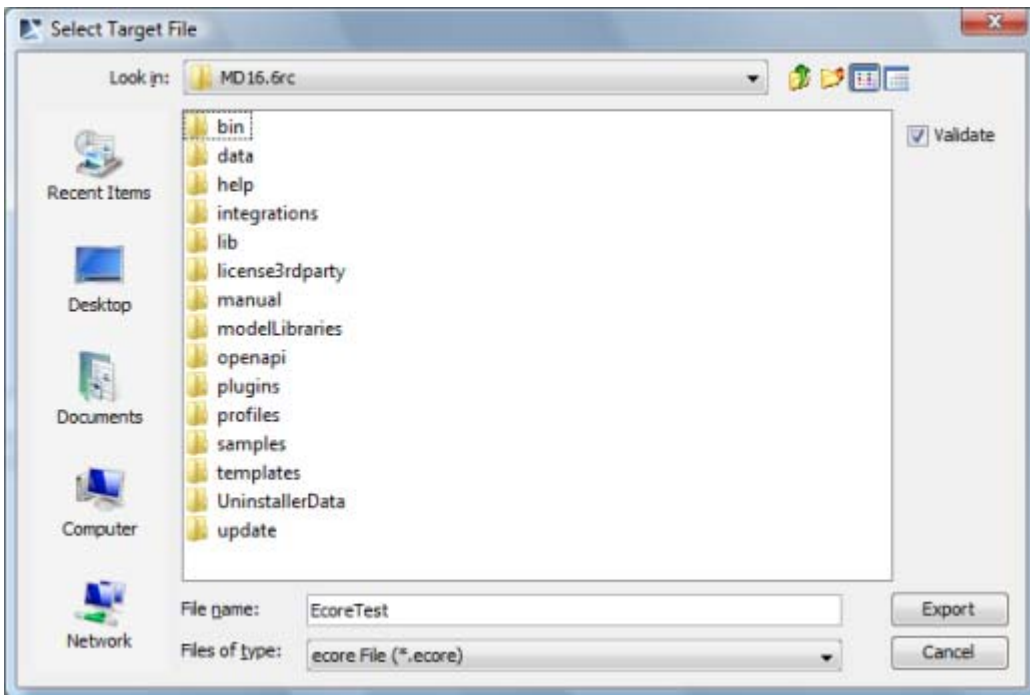


Figure 91 -- Selecting Target File for Ecore Export

To export part of the model, use **File>Export To>EMF Ecore File>Ecore Selection** menu item. Choose packages you want to export in the opened dialog. Afterwards export target file has to be specified - you will be directed to the same file chooser dialog as for exporting whole project.

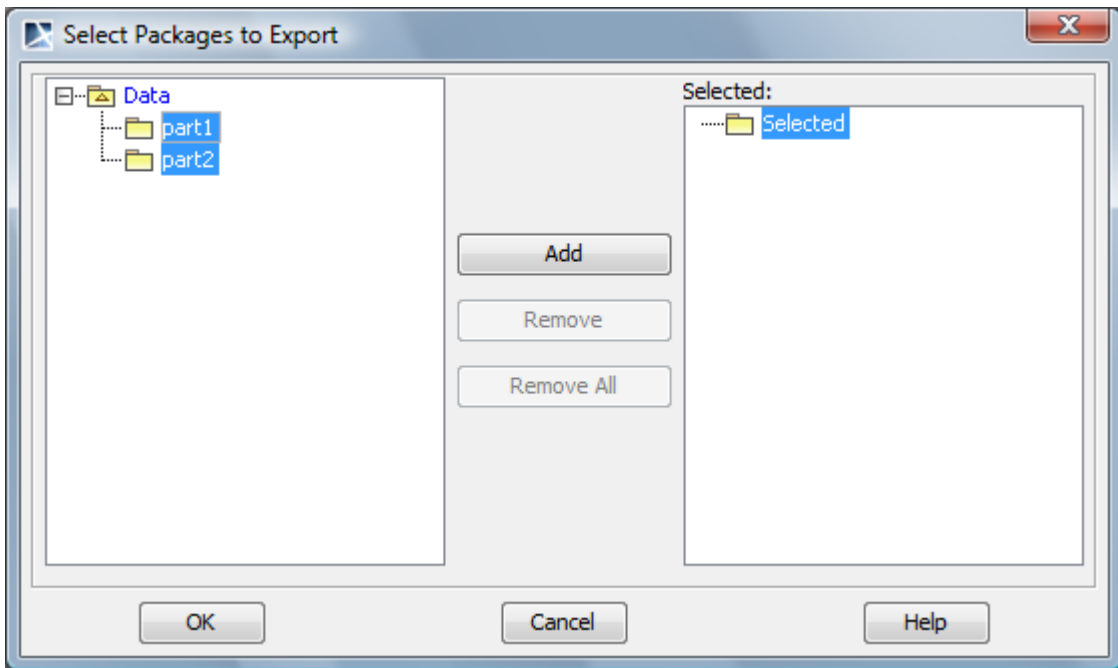


Figure 92 -- Selecting Part of the Model for Ecore Export

You can validate the model being exported - see "Validation" on page 242.

Usages of standard data types from UML - **String**, **Boolean**, **Integer**, **UnlimitedNatural** are exported as standard Ecore data types - **EString**, **EBoolean**, **EInt**, **EInt** correspondingly.

Usages of standard types from MagicDraw profile - **boolean**, **byte**, **char**, **date**, **double**, **float**, **int**, **long**, **short** are exported as Ecore type usages - **EBoolean**, **EByte**, **EChar**, **EDate**, **EDouble**, **EFloat**, **EInt**, **ELong**, **EShort** correspondingly. **void** usage is exported as absence of type.

References to standard Ecore model elements, defined in Ecore model library in MagicDraw (standard datatypes like **EInt**, metaclasses like **EStructuralFeature**) are exported as references to standard elements from Ecore metamodel (Ecore.ecore, resource identifier <http://www.eclipse.org/emf/2002/Ecore>)

Exporting preserves module structure. When you export project, only direct contents of that project are exported - any modules used in the project are not exported. References from project to modules

are exported as Ecore references. Hence the export result is Ecore file/model containing direct contents of the MagicDraw project and referencing other Ecore files/modules wherever there is a reference to module content in MagicDraw project. To export modules to Ecore, you have to open each module as project and trigger Ecore export (note that it is recommended, that modules be exported to Ecore before exporting main project to Ecore - for the reasons described below).

There is one important complication when dealing with projects consisting of modules. It arises because Ecore references, crossing resource boundary (when element in one file references element in another file), are qualified-name based, not id based - as is the case with CMOF, EMOF or UML. When exporting project with references to elements in other projects/modules, export tool must know the full path of elements in the module's Ecore file. This information cannot be obtained normally. The following approach is adopted to resolve this problem:

- When module is exported to Ecore ("module" here means - any project, which has it's contents shared - because any such project can be used as module in another project), information about the paths/qualified names of the elements is stored in special stereotypes/tags of shared packages (EcoreExportServiceInformation stereotype, ecoreExportPath tag). This also means that the project is modified during export. This is a little counterintuitive, but we need to save this information for later usage. You need to save your MagicDraw project afterwards, to preserve these changes.
- When a project is exported to Ecore and this project references elements from module, exporter uses information about element path, which was stored when module was exported to Ecore. If this information is missing (e.g. project is being exported before modules are exported), exporter tool tries to guess the correct element path in the module and gives a warning in the messages window about this.

Because of this complication, it is strongly recommended that modules are exported to Ecore first and projects, using those modules, are exported after that.

Validation

You can check the Validate checkbox to run validation during export - in the same manner as validation is done for CMOF/EMOF export. Validation will be run during export, and all the UML elements, that are not suitable for Ecore export, will be highlighted. Validation just produces warnings about not suitable/skipped elements; having warnings does not preclude the export itself.

You can also run this validation suite at any time while you are developing Ecore model - from main menu, select **Analyze>Validation>Validation** menu item, then choose **Ecore Validation** in the **Validation Suite** field and set the scope accordingly.

Importing

There is currently no possibility to import Ecore files directly. Ecore files could be imported using EMOF import functionality. In Eclipse environment open *.ecore file and save it as *.emof. Eclipse will export Ecore file in EMOF format, which can be opened in MagicDraw. Please note that this indirect way loses some Ecore-specific model details, which are not existent in EMOF.



Working with Standard Profiles

Standard Profiles as System Resources

All profiles and libraries, which are bundled with MagicDraw family products are considered as standard/system resources, which are non-modifiable and are essential for correct tool behavior.

We are highly do not recommend to modify our provided standard profiles and libraries as it could cause problems on version updates and even plugins and core MagicDraw tool malfunctions and model corruptions.

Users will be warned on any intentional or unintentional attempt to modify them:

- Open a profile as a project.
- Use a module in read-write mode.
- Import a module into a project.
- Merge projects.
- any other cases.

Plugin and Profile Versions

Standard profiles are usually upgraded to support newest versions of the specification of the standards they represent (e.g SysML 1.2 or UML 2.2 versions) in every MagicDraw release cycle. The MagicDraw application and plugins code and behaviour are modified accordingly, to reflect these changes.

There is a very high probability that the new version of MagicDraw or plugins can't work with older or newer profile versions and may cause unpredictable behaviour of even model distortions.

(E.g. MagicDraw SysML 16.5 requires to use SysML Profile of version 16.5, as it could malfunction when using SysML Profile from version 16.0 or 16.6).

To protect user from such cases, every MagicDraw project starting from version 16.6 knows which profiles or plugins versions were used to create it and are required to load data correctly.

Every standard profile has version number. Normally it is the same as MagicDraw (or plugin) release version number.

MagicDraw requires to use corresponding version of the profile with corresponding software version so will warn you if your used plugins or profiles are missing or obsolete.

Resource Manager with selected missing resources will be launched automatically, so you will be able to install missing plugins/profiles in few clicks.

If new versions of plugins are not purchased or you simply don't want to install, but need to take a look at the project content, warnings may be ignored and project may be loaded. In this case, proxy elements for missing profile elements will be created to retain missing references. Missing custom diagrams will be loaded as regular UML diagrams or will be restricted for review. Do not save such project! Use it for preview only.

Old projects will be loaded/converted without any warnings if you have newest versions of corresponding plugins and profiles as MagicDraw is always backward compatible.

Standard Profiles in Teamwork Server

Standard/system profiles and modules will not be added into Teamwork Server anymore, because every user has recent versions installed locally. As profiles/modules are non-modifiable, version control is not needed. It solves multiple profiles usage/modification/update issues in teamwork and at the same time increases teamwork performance, as standard profiles will not be transferred via networks.

For more information about Standard Profiles in Teamwork Server, see MagicDraw Teamwork User-Guide.pdf.

5 WORKING WITH DIAGRAMS

This chapter offers an overview of working with diagrams. In general, the topics discussed apply to all supported diagram types.

When working with diagrams it is helpful to keep in mind the following concepts:

- A **shape** refers to a notation of a model element, such as a package, class, state, use case, object, or others.
- A **path** refers to the notation for the various kinds of relationships such as associations, aggregations, dependency, message, and links.
- Both paths and shapes are defined as **symbols**.

In this Section, you will find the following chapters:

1. "Working with Diagrams", on page 246
2. "Table with diagram information", on page 256
3. "Drawing Shapes", on page 257
4. "Drawing the Shapes of the Diagrams", on page 262
5. "Drawing Relationship Paths", on page 263
6. "Inserting a Shape on the Path", on page 266
7. "Creating Relations from the Model", on page 268
8. "Smart Manipulation", on page 269
9. "Selection and Multiple Selections", on page 272
10. "Copying/Pasting Text or Images to Diagrams", on page 275
11. "Tooltip text", on page 279
12. "Using the Grid", on page 291
13. "Layout", on page 292
14. "Showing Diagrams in Full Screen", on page 312
15. "Floating Diagram Window", on page 314
16. "Printing", on page 318

Working with Diagrams

View Online Demo MagicDraw Basics

Diagram Basics

To create a new diagram

- From the toolbar:
Click the desired diagram button on the toolbar. The **Create Diagram** dialog box opens. Enter the name of the diagram and select or create a package where you wish to create your diagram.
- From the **Diagrams** menu:
From the **Diagrams** menu, select the desired diagram. The corresponding **Diagrams** dialog box opens. Click **Add**. The **Create Diagram** dialog box opens. Enter the name of the diagram and select or create a package where you wish to create your diagram.
- From the Browser:
Right-click the desired model element in which you would like to create a diagram and, from the shortcut menu, select **New Diagram**. Type the name for diagram directly in the Browser.
- From the model element **Specification** dialog box:
Open the **Package**, **Profile**, **Model Specification** or other elements **Inner Elements** tab. Click **Create**. The corresponding **Diagram Specification** dialog box opens. In this dialog box, you can define the diagram name, enter documentation, define stereotypes, and add tagged values and/or constraints.

NOTE You may create a diagram in the selected package, model, or profile.

To open a diagram

- From the Browser:
Select **Open** from the diagram item shortcut menu or double-click the diagram item.
- From the **Diagrams** menu:
From the **Diagrams** menu, select the desired diagram. The corresponding **Diagrams** dialog box opens. Click the **Open** button.
- From the Content Diagram (available in the Standard, Professional, Architect, and Enterprise editions only), if a diagram is added to the table of contents or a symbol of a diagram is drawn on the Diagram Pane.

- If the diagram is assigned to a particular model element, double click this model element.

TIP 1!

To load all diagrams that have been created in the project, from the **Diagrams** menu, select **Load All Diagrams**.

TIP 2!

To open the list of diagrams that were most recently closed, from the **View** menu, select **Recently Closed Diagrams** and double-click the diagram you want to open. The F12 key also activates this command.

TIP 3!

In the **General** pane of the **Environment Options** dialog box, you can select a method for loading diagrams while opening a project. Three options are available:

- **Load all Diagrams** – loads all diagrams that exist in the project.
- **Load Only Open Diagrams** – loads only diagrams that were not closed in earlier usages of the project.
- **Do not Load Diagrams** – all diagrams are not loaded and closed after opening a project.

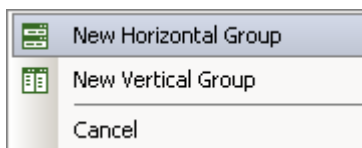
To close a diagram

Click the **Close Diagram** button on the diagram pane or select **Close Diagram** from the diagram shortcut menu.

Define a diagram in the corresponding **Diagram Specification** dialog box.

To split diagrams in new horizontal or vertical group

1. Select the open diagram tab and drag it to the diagram pane. The shortcut menu with commands appears.



2. Select **New Horizontal Group** or **New Vertical Group** to split diagram pane and have more than one diagram opened at the same time.
- Commands can be found in the diagram tab shortcut menu, when so many diagrams are opened that diagram tabs are filled in the toolbar line.

To show the diagram owner on the diagram tab

1. In the package, create diagram.
2. Select the open diagram tab and right-click to open the shortcut menu.
3. Select the **Show Owner** check box. The package name appears on the diagram tab.

Diagram Specification dialog box

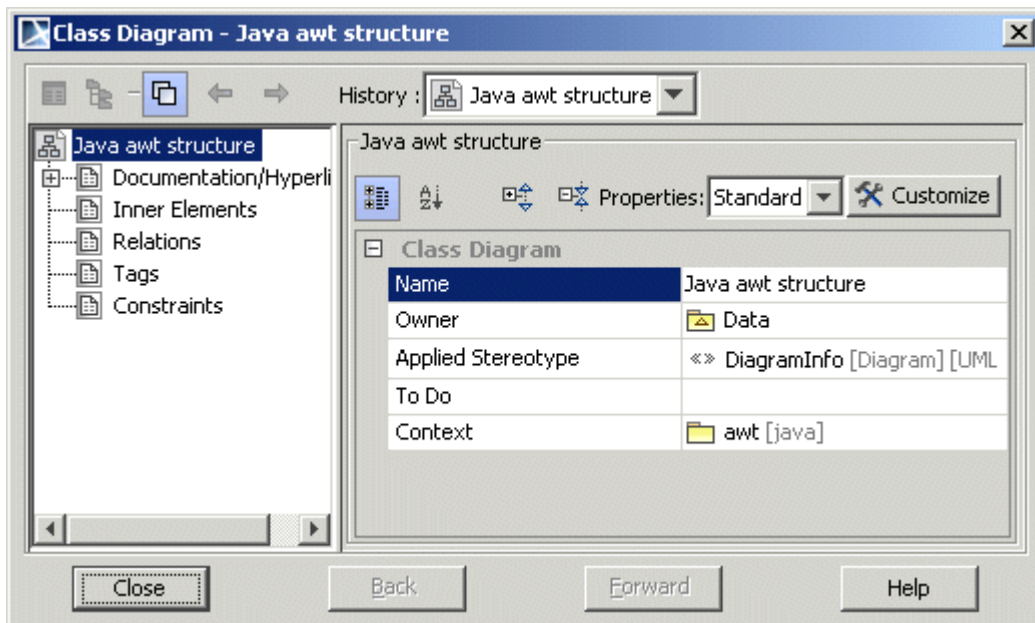


Figure 93 -- Diagram Specification dialog box

Refer to "Specification dialog boxes" on page 325 for information on these specification elements.

To rename a diagram

In the corresponding **Diagram Specification** dialog box, type a new diagram name.

To change diagram context

In the corresponding diagram **Specification** dialog box, near the **Context** box, click the "..." button. Then, the **Select Element** dialog box appears. Select the diagram context.

Diagrams Dialog Box

The **Diagrams** dialog box is used for the following purposes:

- For viewing the owner of the diagram.
- For creating a new corresponding type of diagram.
- For editing the name and other characteristics of the diagram.
- For removing a diagram from the project.
- For opening a diagram.

To open the corresponding **Diagrams** dialog box

From the **Diagrams** menu, select one of the diagrams. Depending on the type of diagram, the dialog box that opens has a corresponding title.

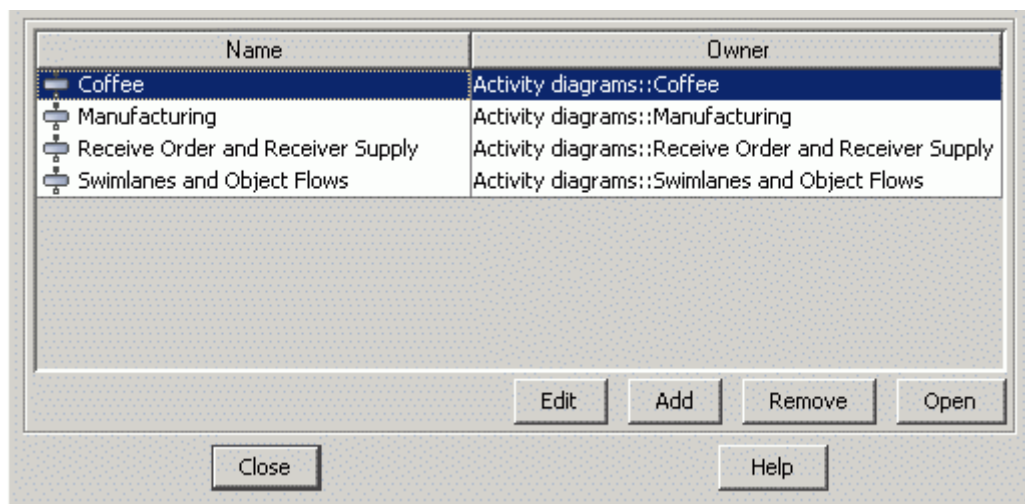


Figure 94 -- Diagrams dialog box

The **Diagrams** dialog box contains the following elements:

Element name	Function
Name	The names of all created corresponding diagrams in the open project.
Owner	The name of the package that owns the diagram.

Element name	Function
Edit	The Diagram Specification dialog box opens. Type the diagram name, select a package, and click OK .
Add	Creates a new diagram. The Create Diagram dialog box opens. Type the diagram name, select a package, and click OK .
Remove	Deletes the selected diagram.
Open	Opens the selected diagram.
Close	Saves all actions performed during the session and exits the dialog box.
Help	Displays the MagicDraw Help.

Diagram Properties

Customize the diagram style (color, grid) in the **Diagram Properties** dialog box.

To open the **Diagram Properties** dialog box

- Select **Diagram Properties** from the diagram shortcut menu. The **Properties** dialog box opens.
- From the **Edit** menu, select **Symbol**, and then select **Diagram Properties**.

- Press SHIFT+ENTER.

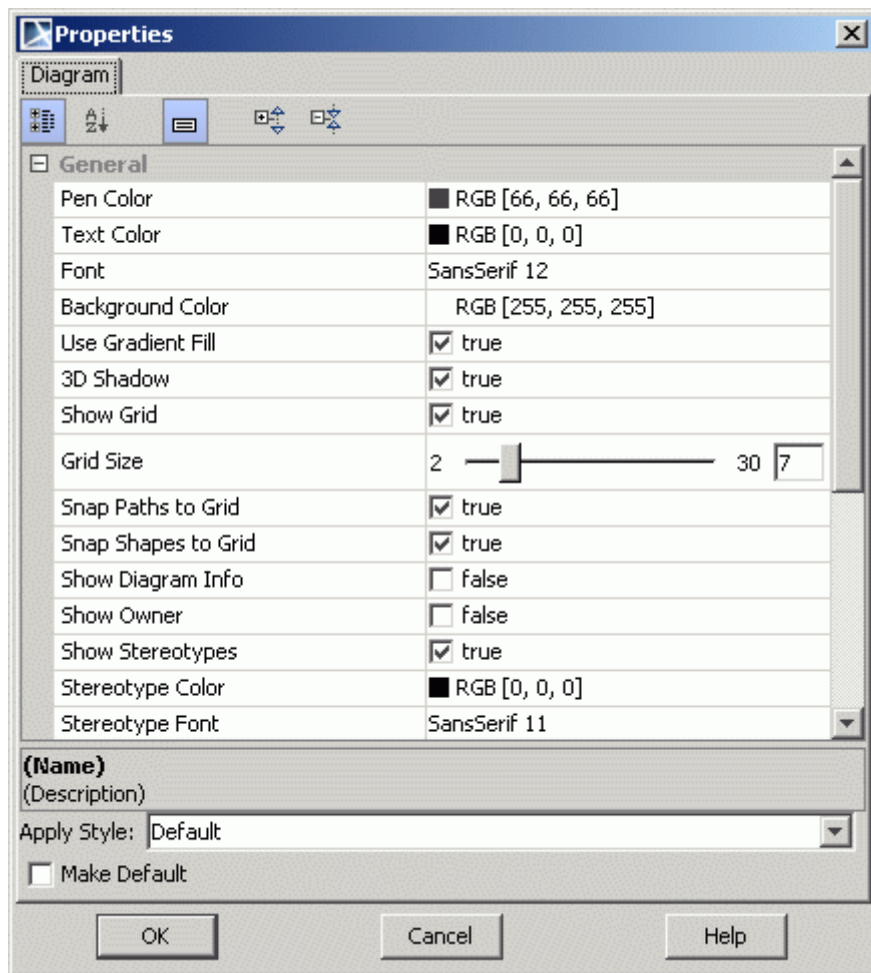


Figure 95 -- Diagram Properties dialog box

The **Diagram Properties** dialog box contains the following elements:

Element name	Function
Pen Color	Change the color of paths or diagram frame line.
Text Color	Change the text color of the diagram frame name.
Font	Change the font style of the text.

Element name	Function
Background Color	Set the diagram background color. Click the '...' button. The Color dialog box opens. Set the color in one of three different ways: using Swatches , HSB , or RGB tabs.
Use Gradient Fill	Sets the gradient for the shapes fill color.
3D Shadow	Displays 3D shadow on symbols.
Show Grid	Shows the grid on the Diagram pane.
Grid Size	Sets grid size from 2 to 30.
Snap Paths to Grid	Uses the grid on the diagram for drawing paths.
Snap Shapes to Grid	Uses the grid on the diagram for drawing shapes.
Show Message Numbers (Available only for Activity, Communication, and Sequence diagrams)	Displays message numbers on the diagram.
Use Advanced Numbering (Available only for Activity, Communication, and Sequence diagrams)	Displays more detailed message numbering on the diagram.
Show Diagram Info	Displays a table on the diagram that contains information about the diagram (Diagram name, Author, Creation date, Modification date, etc.). You can specify what information you want to include in the Project Options dialog box (Options menu-> Project).
Show Owner	Displays diagram owner on the diagram tab.
Show Stereotypes	Displays stereotypes on the diagram symbol.
Use Stereotype	Utilizing the Use Stereotype property, you may choose to display context stereotype or diagram stereotype in the diagram frame and on the diagram shape. For more information about the Use Stereotype property, see "To display the context stereotype instead of the diagram stereotype on the diagram frame" on page 256.
Stereotype Color	Changes the color of the stereotyped text label.

Element name	Function
Stereotype Font	Changes the font style of the stereotyped text label.
Diagram Orientation	Available for Activity and Business Process diagrams for correct rectilinear path braking and drawing paths between shapes from side to side, or from bottom to top shape borders.
Apply Style	Selects the predefined style.
Make Default	Sets the specified style as default.
OK	Saves changes and exits the dialog box.
Cancel	Exits the dialog box without saving changes.
Help	Displays MagicDraw Help.

Diagram name and its context name synchronization

The diagram name and its context name are synchronized automatically.

For example, create an Activity diagram. Type a name for the Activity diagram, for example, Receive. The name of the Activity automatically changes to Receive. And conversely - change the name of the Activity and the Activity diagram name will be changed automatically. This is synchronization of a diagram name and its context name:

Synchronization works in the following cases:

- Activity and Activity diagram inside.
- Interaction and Communication or Sequence diagram inside.
- (Protocol) State Machine and (Protocol) State Machine diagram inside.
- Class and all available inner diagrams inside.

To turn off the synchronization clear the **Synchronize the diagram name with it's context name** check box in the **Environment Options** dialog box, **General** branch, and **Editing** group.

NOTE If the second diagram will be created in the branch, diagram names will not be synchronized.

Diagram Frame

According to UML specifications, the diagram may have a diagram frame. The frame is primarily used in cases where a diagrammed element has graphical border elements (like ports, gates).

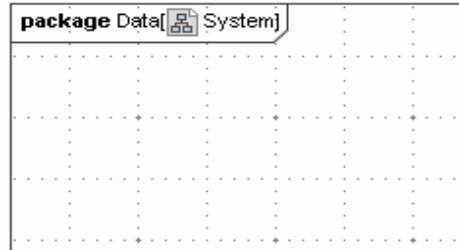
Beginning with MagicDraw version 12.0, each UML diagram has a contents area. The diagram frame is primarily used in cases where the diagrammed element has graphical border elements (like ports for classes and components, entry/exit points on statemachines).

By default, the diagram frame is displayed in the diagram pane when a new diagram is created. The frame is rectangle for all diagrams, except State and Activity. State machine and Activity diagram frames have rounded corners.

The frame can be resized manually by selecting and dragging the corners or borders.

To show/hide the diagram frame

From the diagram pane shortcut menu, select/clear the **Show Diagram Frame** check box.



To change the diagram frame properties

1. From the **Options** main menu, select **Project**. The **Project Options** dialog box opens.
2. Select the **Diagrams** group and change the properties in the right window, **Diagram Frame** properties group.

5 WORKING WITH DIAGRAMS

Diagram Frame

- From the diagram frame shortcut menu, select **Diagram Properties** or press SHIFT+ENTER to open the **Properties** dialog box. Change the properties in the **Diagram Frame** properties group.

Property name	Function
Show Diagram Frame	Displays the diagram frame on the diagram pane.
Show Abbreviated Types	Shows full/abbreviated diagram keyword type on the diagram frame header.
Show Diagram Name	Shows the diagram name and icon in the diagram frame header.
Show Parameters	Diagram context element parameters are displayed in the diagram frame header.
Show Context Name	Diagram context element name is displayed in the diagram frame header.
Show Context Type	Diagram context element type is displayed in the diagram frame header.
Show Diagram Type	Shows the diagram type in the diagram frame header.
Show Context Kind	Shows context kind, which is a keyword predefined in UML (e.g. package, class, activity) in the diagram frame header.
Autosize	Adjusts the size of the diagram frame to the contained information so that it uses minimum space.

To hide the icon on the diagram frame

1. Invoke the diagram **Properties** dialog box.
2. Change the **Show Stereotypes** property value to **Text**.

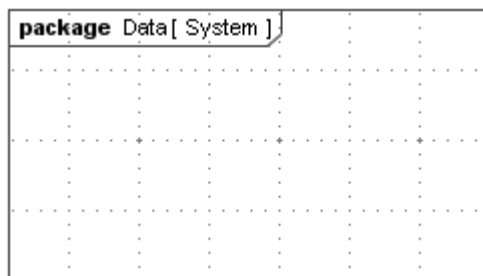


Figure 96 -- Diagram Frame with hidden diagram icon

To display the context stereotype instead of the diagram stereotype on the diagram frame

In the diagram **Properties** dialog box, **Diagram Frame** properties group, select one of the **Use Stereotype** property options:

- **Diagram.** Diagram stereotype is displayed in the diagram frame and on the diagram shape. This is the default property.
- **Context.** Context stereotype is displayed instead of the diagram stereotype in the diagram frame or on the diagram shape.

NOTES

- For more information about the context of a diagram, see “To change diagram context” on page 248.
- The **Use Stereotype** property is associated with the **Show Stereotypes** property. If you want to change the stereotype and its icon visibility use the **Show Stereotypes** property. For more information about the **Show Stereotypes** property, see “Changing the stereotype display mode” on page 809.

Table with diagram information

On the diagram, you may display a table containing various diagram details: its name, author, status, the dates it was created and modified, etc. By default, the Diagram info table is displayed at the right top corner of the diagram frame, but you can drag and drop it to any other position on the diagram.

The table includes the following fields:

- Diagram name
- Author
- Creation date
- Modification date
- Other available tag definitions

To show the table containing the diagram information

1. From the diagram shortcut menu, select **Show Diagram Info**.
2. The table with the predefined information will be displayed on the diagram.

To define information that will be included in the table

1. From the diagram info table shortcut menu, select **Customize**, or from the **Options** menu, select **Project**. The **Project Options** dialog box opens.
2. Open the **Diagram Info** pane.
3. In the **Source** pane, select the type of information you want to include in the table: **Standard Mode** or **Custom Mode**.
4. **Standard Mode** contains the following fields that will be shown in the table: Author, Creation date, Modification date, and all other tag definitions that can be assigned to the diagram.
In the **Custom Mode** field, you may create your own table or any other object in HTML.
5. Preview the selected table or other created object in the **Preview** pane.

Drawing Shapes

[View Online Demo](#)

MagicDraw Basics

To draw a shape on the Diagram pane

1. Click the shape button on the diagram toolbar, or press the appropriate shortcut key for the shape you wish to draw (the button remains pressed). For a detailed description of diagram toolbars, see “Toolbars” on page 102.

2. Click the desired location on the diagram pane. The new shape is placed on the diagram pane at the point you click.

OR

- Click the shape button on the diagram toolbar, hold down the left mouse button, and drag shape from the toolbar to the diagram. The new shape is placed on the diagram pane at the point you will release the mouse.
- Create the desired model element in the Browser tree. From the created item shortcut menu, select **Create Symbol** or drag and drop the selected model element to the diagram pane.

To draw a number of shapes on the diagram pane

1. Click the shape button on the diagram toolbar, or press the appropriate shortcut key for the shape you wish to draw (the button remains pressed.)
2. Click the **Sticky** button on the diagram toolbar (shortcut key is Z.)
3. Click the desired location on the diagram pane. The new shape is placed on the diagram at the point you click (the button remains pressed.)
4. Click the next location on the diagram pane. The next shape is placed on the diagram pane. Repeat this until you draw the desired number of shapes.
5. To undo the shapes, click the **Sticky** button on the diagram toolbar (shortcut key is Z).

To draw a shape for the selected item in the Browser tree

1. Activate a diagram on which you wish to draw a shape.
2. From the Browser tree, select an item you wish to draw.
3. From the item shortcut menu, select **Create Symbol** or drag and drop the selected model element onto the diagram pane.

To specify the name of the shape (when it is allowed)

1. Double-click the shape or select **Specification** from the shape shortcut menu. The corresponding **Specification** dialog box opens.
2. Type the shape name in the **Name** text box and click **Close**.
 - Type the shape name directly on the selected shape on the Diagram pane.
 - Type the shape name after slowly double-clicking the shape in the Browser tree.

To create several shapes with the same data

- From the shortcut menu of the desired item in the Browser tree, select **Create Symbol**. Or, drag and drop the selected model element onto the diagram pane.
 - Type the same name for multiple shapes directly on the shape after the text cursor appears in that area.
1. Specify a shape name.
 2. Draw another shape of the same kind on the Diagram pane.
 3. Click the shape in the name area. The list of existing shape names appears.
 4. Select a name for the shape from the list.

NOTES

- These shapes will contain identical data.
- Auto completion for entering names is available for all elements.

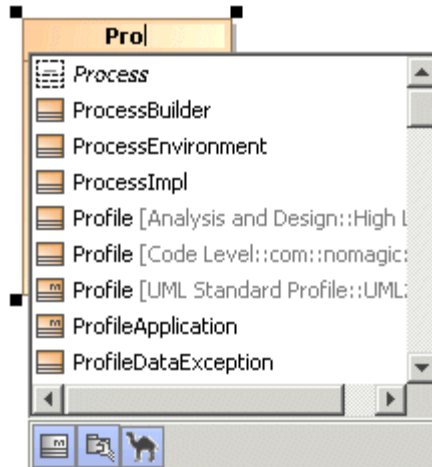
IMPORTANT

If you attempt to enter an existing name in the corresponding **Specification** dialog box, an error message alerts you to the existence of the current name of the shape. You may not specify a name for a new shape if another shape of the same name and kind is already present in the package.

To use autocompletion when typing the element name

- On the diagram pane, click on the element and then click on the element name area. The autocompletion list with already created elements appears.

- Type the first letter(s) of your searched element and the list is reduced according to the letters you typed. For, example, if you are searching for *Profile* class, type the *Pro* letters and all classes, which begins with *Pro* will be shown in the list.,



You can also press the **Ctrl+Space** or **Ctrl+Backspace** to invoke the autocompletion list.

Filters in the autocompletion dialog allow the filtering of rarely used items as "metaclasses" and "elements from profiles". This allows comfortable and clear usage of the autocompletion dialog for modeling without active usage of elements from profiles and it increases modeling speed.

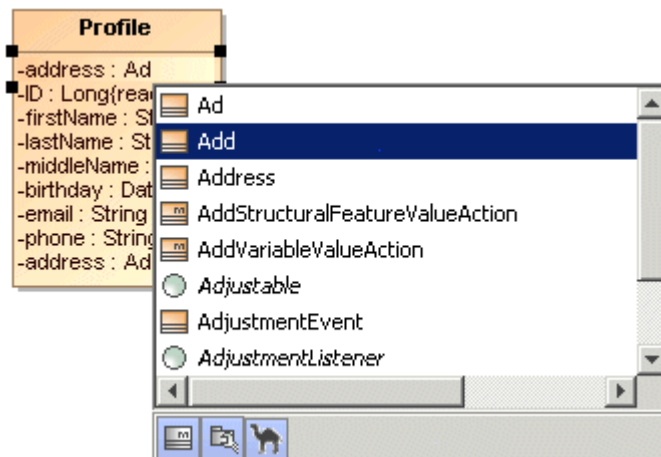
At the bottom of the drop-down list box, you will find buttons to perform filtering:

- **The Auto completion includes metaclasses button.** When pressed, the list of available elements, element types, or stereotypes includes metaclasses (in MagicDraw metaclasses are placed in the *UML Standard Profile*) appears.
- **The Auto completion includes elements from profiles and modules button.** When pressed, the list of available elements, element types or stereotypes includes elements, which are placed in modules appears. (**Note!** This option toggles all profiles except the *UML Standard Profile*.)
- **The Auto completion uses camel case button.** When pressed, you may search for elements via the capital letter patterns. For example, instead of typing *ArrayIndexOutOfBoundsException* you may type *AIOOBE*.

To assign an existing type or to create a new type to an element

You may quickly assign a type for the attribute, operation, parameter, instance, and lifeline using the autocompletion list:

1. On the diagram pane in the element name area type ":" and now you may assign the element its type:
 - Type the name of a non-existent element. A new class and the assigned type are created in the project.
 - Type the name of an already existing classifier and it will be assigned as the type.
 - If in the project two classifiers exist with the same title after the classifier name is typed, the Select Classifier dialog box opens. Select the element you want to assign as type.
2. On the diagram pane in the element name area type ":" and press **Ctrl+Space** or **Ctrl+Backspace**. The list of possible elements to assign opens. To find the element in the list by name, type its name.



To delete the selected model element or symbol

- From the **Edit** menu, select **Delete** (both data and symbol are deleted.)
- On the main toolbar, click **Delete** (both data and symbol are deleted.)
- Press CTRL+D keys (both data and symbol are deleted.)

- Press Delete key (only the symbol is deleted, leaving the data intact.)

NOTE

When deleting paths by pressing the DELETE key, the **Delete Data?** message appears. To delete the relationship data from the model, click **Yes**.

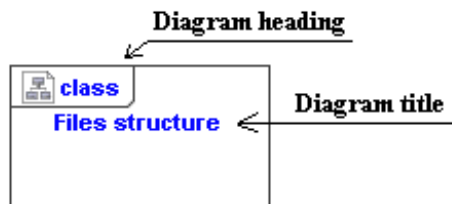
When you use other methods to delete relationships, the relationship data is automatically deleted.

Drawing the Shapes of the Diagrams

UML diagrams are graphical representations of parts of the UML model. You can show any diagram shape in all types of diagram.

To display a diagram shape on the diagram pane

1. Select the diagram in the Browser.
2. Drag the diagram and drop it in the diagram pane. The diagram symbol is displayed on the diagram pane. The diagram type is written in the diagram heading - class, use case, etc.



To display stereotypes, tags, and constraints on the diagram shape

1. From the diagram shape shortcut menu, open the **Symbol Properties** dialog box.
2. Select the **Show Stereotypes**, **Show Constraints**, or **Show Tagged Values** check boxes.

In the diagram heading you cannot display the full diagram type name, but an abbreviated diagram name.

To display the abbreviated diagram type

1. On the diagram pane select the diagram shape.
 2. In the shortcut menu, select the **Show Abbreviated Type** check box.
- or-
- From the diagram shape shortcut menu, open the **Symbol Properties** dialog box and select the **Show Abbreviated Type** check box.

The list of abbreviated diagram types is presented in the following table:

Diagram name	Abbreviation
Use Case	uc
Communication	comm
Sequence	sd
State Machine	stm
Activity	sct
Implementation	impl
Composite Structure	cs

Drawing Relationship Paths

[View Online Demo](#) MagicDraw Basics

To create a path between shapes

1. Click the appropriate path button on the diagram toolbar for the relationship you wish to draw. For a detailed description of the diagram toolbars, see “Toolbars” on page 102.
2. Click the first (source) shape of the path.
3. Drag the path to the second (target) shape of the path and drop it there.

To create a number of paths

1. Click the appropriate path button on the diagram toolbar.
2. Click the **Sticky** button on the Diagram toolbar (shortcut key is Z.)
3. Click the first (source) shape of the path.
4. Drag the path to the second (target) shape of the path and drop it there.
5. Click the first (source) shape of the path to draw the next path.
6. Drag the path to the target shape and drop it there. The new path is created between the two shapes. Repeat this until you create the desired number of paths of that type.
7. Click the **Sticky** button on the Diagram toolbar (shortcut key is Z.)

To change a path appearance style

1. Select the path.
2. Right-click the path or select **Path** from the **Edit** menu, and select the commands you need:
 - To set the path as rectilinear, oblique, or bezier, select **Path Style**.
 - To select a path style, select **Change Path Style** (shortcut keys CTRL+L.)
 - To reset path labels to the default position, select **Reset Labels Positions**.
 - To remove all angles of the path, select **Remove Break Points**.

NOTES

- Every diagram has the Manipulation Highlighting feature. When drawing a path between two model elements, you will see that those shapes are bordered with a red or blue rectangle. The red color indicates that the path may not be drawn between these shapes. Blue rectangle allows a path to be drawn.
- Remove the manipulation highlighting in the **Environment Options** dialog box, **Diagram** section, **Edit** group. For more information, see “Setting Environment Options” on page 129.
- For drawing a path, you can use smart manipulations menu, which appears near the element symbol. Select a path and drag it to the target shape.

To make the path corners rounded

1. Select the path.

2. From the paths shortcut menu, select the **Rounded Corners** check box.
 - Right-click the path and select **Symbol(s) Properties** and in the **Properties** dialog box, set the **Rounded Corners** property to *true*

To create line jumps

Line jumps represent an intersection of lines. If you have a large diagram with lots of intersecting paths, line jumps make the diagram easier to understand.

By default line jumps are not displayed. You can configure a diagram to display line jumps in the following way:

1. From the diagram shortcut menu, choose **Diagram Properties**. The **Properties** dialog box appears.
2. Change the **Add Line Jumps To** property.

The **Add Line Jumps To** property has the following options:

- *None*. Line jumps are not displayed. (Default value)

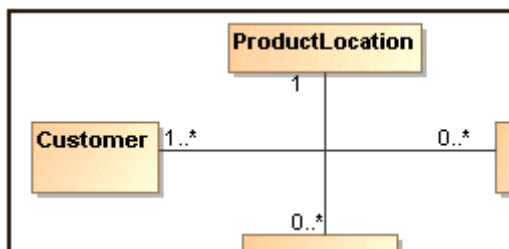


Figure 97 -- Diagram with no line jumps

- *Horizontal Line*. Line jumps are displayed on horizontal lines.

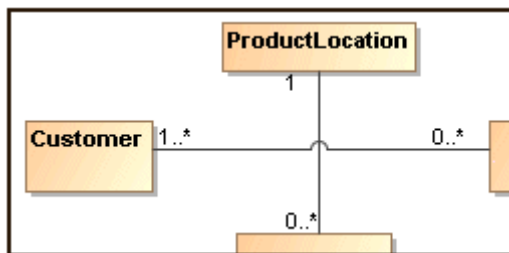


Figure 98 -- Diagram with horizontal line jumps

- **Vertical Line.** Line jumps are displayed on vertical lines.

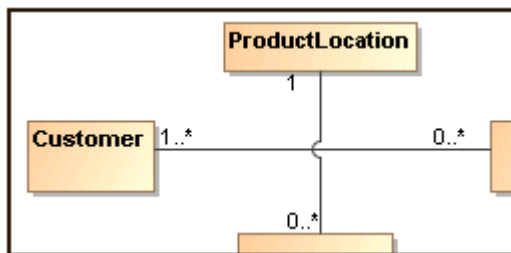


Figure 99 -- Diagram with vertical line jumps

Changing the **Add Line Jumps To** property for a particular diagram from within the diagram **Properties** dialog box, will change line jumps for the current diagram only. For more information about how to change line jumps for the whole project, see “Style Engine” on page 346 .

Inserting a Shape on the Path

NOTE: This functionality is available in the State and Activity diagrams.

In the State and Activity diagrams you may split a path into two paths, by drawing a symbol on it. This is valid for Transition / Control Flow / Object Flow relationships and allowed to connect with these path elements.

To insert a new shape splitting path on the diagram pane

1. Select the symbol you want to insert or click the diagram toolbar button to create a new one.
2. Drag it on the path. The path is highlighted in blue.
3. Drop the symbol. A Message dialog box appears asking if you want to insert the symbol on the path.

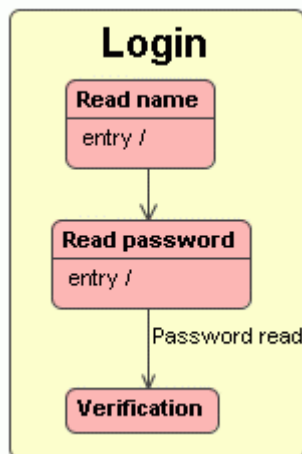
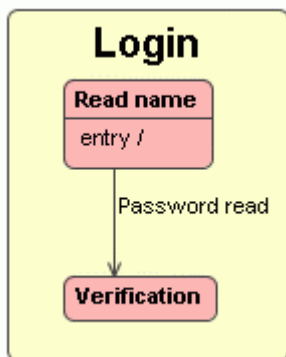
Possible solutions:

- **Before <path type>.** Symbol is inserted before the path. It means a new path is created, then the dropped element symbol is drawn and then the existing path is drawn. For example:
Password read transition is drawn from *Read Name* state to *Verification* state. If you want to

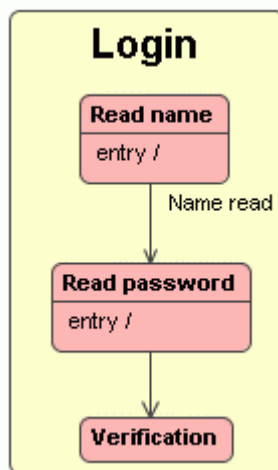
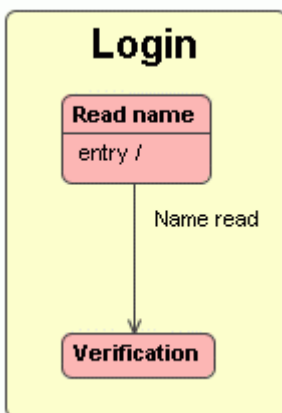
5 WORKING WITH DIAGRAMS

Inserting a Shape on the Path

insert *the Read password* state before the *Password read* transition, drop the *Read password* state on the transition and in the open dialog, click the **Before Transition** button.



- **After <path type>.** Symbol is inserted before path. It means, the existing path is created, then the dropped element symbol is drawn, and then a new path is drawn. For example: *Name read* transition is drawn from *Read Name* state to *Verification* state. If you want to insert the *Read password* state after the *Name read* transition, drop the *Read password* state on the transition and in the open dialog, click the **After Transition** button.



- **Do not insert.** Action is cancelled and the dialog is closed.

Select the Remember my choice check box and the next time an element will be inserted before or after the path, depending on your selection made this time.

Creating Relations from the Model

The main purpose of this functionality is to allow connecting and create traceability, according to UML, among elements, which are not from the same diagram. In other words, to link elements from a model without the need to place them in the same diagram.

Advantages of this implementation:

- Working time is saved on creating a diagram just to link elements for traceability.
- Some elements cannot be added to the same diagram and linked (elements from Behavior diagrams - Actions, States, Lifelines cannot be added to Static diagrams), this feature will allow the linking of such elements.
- Capability to relate multiple elements to a single element quickly. This is the usual case when a single element is represented with many elements in different abstraction levels or domains.
- The allocation relationship can provide an effective means for navigating the model by establishing cross relationships and ensuring the various parts of the model are properly integrated. For example, activity allocation to a bloc in SysML.
- Another example is creating an abstraction relationship with a stereotype <<trace>> between the model elements or sets of model elements that represent the same concept in different models.
- Smart manipulators allow connecting to any existing element quickly. Also, any Relation used with an element has a smart manipulator included. Draw any relation from any element and you will be able to select the existing target element from the browser.

To create a new relation for an element

1. From an element shortcut menu in the browser, select **New Relation** and then select the desired link from the group of **Outgoing** or **Incoming** relations. The **Create New <relation name> To (From)** dialog box opens.
 2. In the model element tree, select an element to (from) which you want to create a relation. Click **OK**. The link will appear in the Browser. Type the name or leave it unnamed.
- or-

1. In the element **Specification** dialog box, select the **Relations** group.
 2. Click the **Outgoing** or **Incoming** button and then select the desired link from the list. The **Create New <relation name> To (From)** dialog box opens.
 3. In the model element tree, select an element to (from) which you want to create a relation. Click **OK**. The link will appear in the **Relations** group.
- or-
1. On the diagram pane, select an element and then select the desired link from the smart manipulator relations list that opens.
 2. Right-click to open the target element list and select the **Select From Model** command. The **Create New <relation name> To** dialog box opens.
 3. In the model element tree, select an element to which you want to create a relation. Click **OK**. The link will be drawn on the diagram pane.

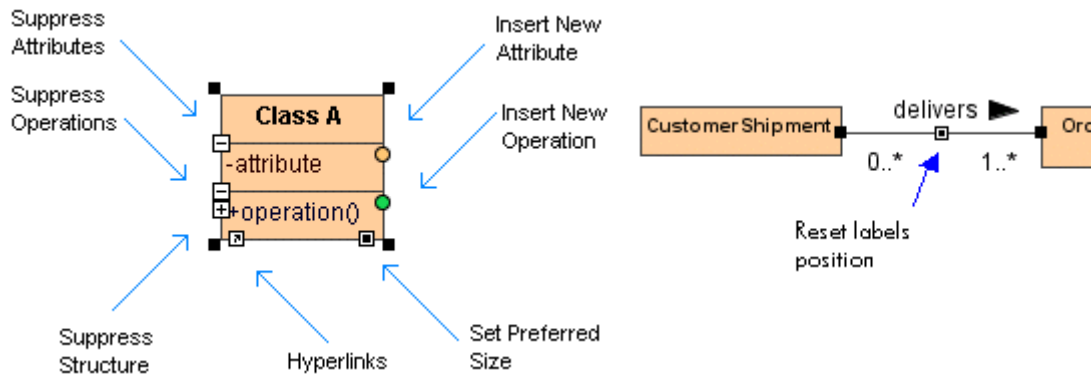
Smart Manipulation

[View Online Demo](#) Smart Manipulators

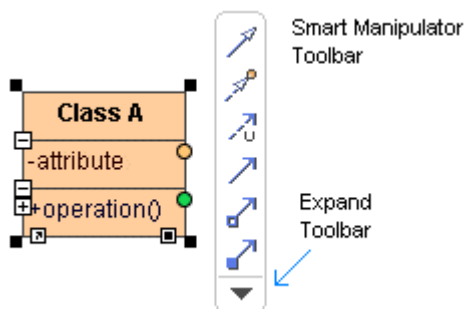
Smart Manipulation is a feature designed to make working with MagicDraw even easier. Use Smart Manipulation to suppress attributes and operations, set an auto-size option, reset a label position on a path, and draw relationships with most commonly used elements. MagicDraw offers varying smart mechanisms depending on the shapes involved.

There are two types of Smart Manipulators:

1. Small buttons are displayed within the symbol on the diagram pane. .



2. Smart Manipulator toolbar, which appears when elements are selected on the diagram pane.



In the Smart Manipulator toolbar, smart manipulators are divided into *standard* and *extra* modes. You can toggle between these two modes by clicking the **Expand** button - the arrow symbol - located at the bottom of every Smart Manipulator toolbar. The program remembers your mode choice and displays it for all elements.

Use the Smart Manipulators toolbar to quickly perform simple actions and create new elements.

To create a new element connected to a particular element

1. Select a symbol on the diagram pane. The Smart Manipulator toolbar appears. In the toolbar, select the relationship you want to draw. The drawing of the selected relationship is initiated and the mouse cursor displays the new element which will be created.

For example, create a class symbol. In the class Smart Manipulator toolbar, select the Directed Association relationship for drawing. The drawing of this Directed Association relationship is initiated and the mouse cursor displays a class icon. Click the left mouse button. The element displayed on the mouse cursor is created together with the relationship.

2. Use the Smart Manipulator toolbar to select which element you want to draw at the other end of relationship. In the toolbar, select the relationship and then click the right mouse button. The list of elements available for creating appears. Select the element from the list and it will be created.

TIP! To create a path breakpoint use the following keyboard combination: Ctrl key + Mouse click.

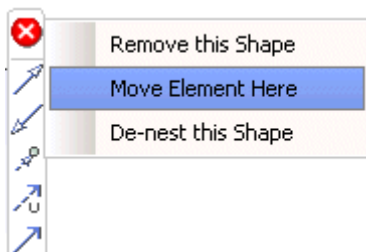
TIP! To cancel the drawing of an element, press **Esc**.

The Autosize option is automatically added for all shapes. To hide smart manipulation

1. From the **Options** menu, select **Environment**. The **Environment Options** dialog box opens.
2. In the **Diagram** pane, **Smart Manipulators** group, clear the **Show Smart Manipulation** check box and click **OK**.

To solve the detected symbol ownership problem

1. Select the element from the diagram pane, which is drawn on an incorrect ownership (which is highlighted in red). The Smart Manipulator toolbar appears.
2. Click the red button, which is at the top of the Smart Manipulator toolbar. The menu with the possible problem solving solutions appears.



For more information see “Resource Manager” on page 497.

Selection and Multiple Selections

To select a shape

- Click the desired shape on the Diagram pane.

To deselect the selected shape

Click outside the shape on the Diagram pane.

To select all shapes of the same type

Press ALT and click the shape. All shapes of the same type are selected.

To select all shapes on the diagram

From the **Edit** menu, select **Select All** (shortcut keys CTRL+A.) To make multiple selections

1. Click the shape on the Diagram pane.
2. Hold down the SHIFT key and click another shape. Repeat until you select the desired number of shapes.
 - Drag the cursor diagonally across the area you wish to select. All shapes in the selected area will be selected.

To select a group of shapes

To select a group of shapes drag the cursor diagonally across the area you wish to select. This is a simple and fast way to select a group of shapes on the diagram.

See the sample of the rectangular selection in Figure 1.

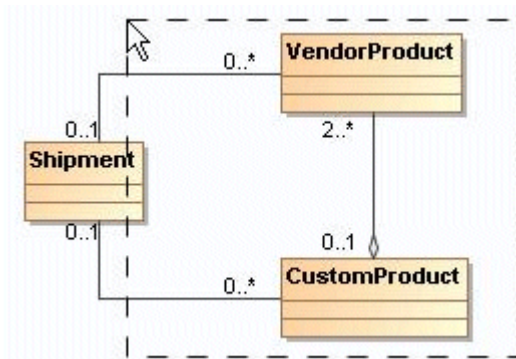


Figure 100 -- Rectangular selection

After the selection process represented in the Figure 1, the following shapes are selected, as shown in Figure 2.

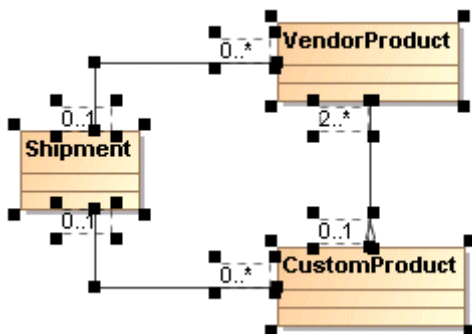


Figure 101 -- Rectangular selection result with partial selection coverage mode

The following rectangular selection modes are available:

1. *Partial coverage.* Symbols which are only partly covered with the rectangular selector are selected. See Figure 2. After the selection process represented in Figure 1, the class *Shipment* is selected by the rectangular selector, even though it was only partially covered.
2. *Complete coverage.* Only those symbols that are fully covered by the rectangular selection process will be selected. See Figure 3. For example, after the selection pro-

cess shown in the Figure 1, the class *Shipment* and the associations are not selected because these symbols were not fully covered.

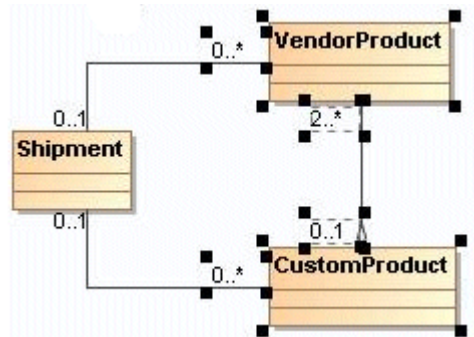


Figure 102 -- Rectangular selection result with complete coverage selection mode

Default selection mode is Partial coverage.

To quickly change the group selection mode from Partial coverage to Complete coverage mode or conversely:

- Press the **Ctrl** key and then drag the cursor diagonally across the area you want to select.

To change the group selection mode for the whole project:

- In the main diagram toolbar press the **Complete coverage mode for group selection** button (see Figure 103 on page 275).

OR

1. From the **Options** main menu, select **Environment**. The **Environment Options** dialog box appears.
2. In the **Diagram** branch, **Symbols Manipulation** group, change the property of the **Group selection mode** option.

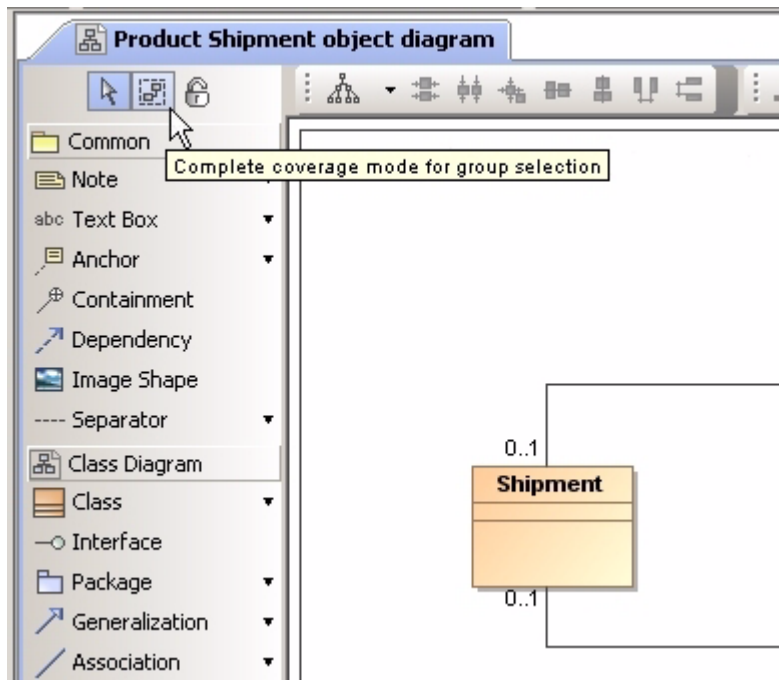


Figure 103 -- The Complete coverage mode for group selection button

New! Copying/Pasting Text or Images to Diagrams

It is now possible to copy and paste text or images to a diagram. A text box and an image shape will be available for the copied text or image. MagicDraw supports HTML and plain text, .gif, .jpg, .svg, and .png image file types. To copy and paste text or image:

1. Copy text or image(Ctrl+C).
2. Open a MagicDraw diagram.
3. Paste the copied text or image (Ctrl+P). The **Paste Special** dialog will open (Figure 104 on page 276).

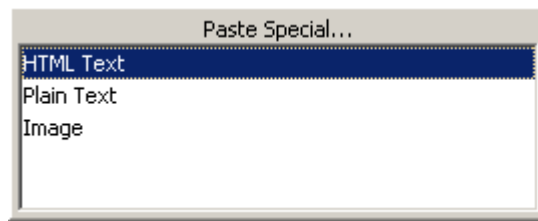


Figure 104 -- The Paste Special Dialog

NOTES

- This copy and paste feature functionality has been extended, allowing you to drag and drop from other applications such as Web browsers.
- The **Paste Special** dialog will open only if the clipboard contains any text or images, text, or HTML text formats.

Nesting Image Shapes

You can now drag an image to any elements in a diagram as nested a element (Figure 105 on page 277).

To drag an image to an element:

1. Select an image in the diagram pane.
2. Drag it to the image shape.

Dragged images will be nested by the following elements: Package, Model, Subsystem, Instance, Node, Part, Combined Fragment, Composite State (State diagram), Interruptible Activity Region, Structured Activity Node, Expansion Region, and Conditional Node (Activity diagram).



Figure 105 -- Samples of Images Nested to the Products Package and Server Component

Dragging, Copying, Cutting, and Pasting

Move a shape to another location on the diagram pane by dragging-and-dropping.

To drag multiple selected symbols

Select the symbols and drag them to the desired area on the diagram pane.

To copy a shape using dragging-and-dropping

Hold down the CTRL key while dragging the selected shape to the area where you wish to make a copy.

Drag and drop items from the browser to the diagram pane

1. In the Browser tree, select the created model element you wish to draw.

2. Drag it to the desired location on the diagram pane and drop it there.

NOTES

- You may select several model elements and draw them on the diagram pane.
- If the selected model elements are not compatible with the open diagram, you will not be allowed to draw those model elements.

To copy/cut and paste the selected shape on the diagram

1. From the **Edit** menu, select **Copy/Cut** (shortcut keys CTRL+C/CTRL+X.)
2. From the **Edit** menu, select **Paste** (shortcut keys CTRL+V.)

NOTE:

You may copy/paste many (but not all) model elements among various diagrams.

To paste one or more copied model elements by creating new data and symbols

From the **Edit** menu, select the **Paste With New Data** command (shortcut keys CTRL+E.)

To copy the whole diagram and paste it to MS Office or other application

1. Select or deselect all model elements on the diagram.
2. From the **Edit** menu, select **Copy as BMP Image**, **Copy as EMF Image**, **Copy as JPG Image**, or **Copy as PNG Image** (shortcut keys CTRL+SHIFT+B, CTRL+SHIFT+E, CTRL+SHIFT+J, or CTRL+SHIFT+P)
3. Open the desired application and paste the copied diagram.

To copy the selected model elements and paste to MS Office or other application

1. Select the desired model elements on the diagram pane.
2. From the **Edit** menu, select **Copy as BMP Image**, **Copy as EMF Image**, **Copy as JPG Image**, or **Copy as PNG Image** (shortcut keys CTRL+SHIFT+B, CTRL+SHIFT+E, CTRL+SHIFT+J, or CTRL+SHIFT+P).
3. Open the desired application and paste the copied model elements.

TIP!

You can drag and drop source code files from the native file manager to a MagicDraw Code Engineering Set.

NOTE

You can copy or cut and paste the text only when using the shortcut keys CTRL+C or CTRL+X and CTRL+V. When you use the buttons or commands, the whole element is copied/cut and pasted.



Tooltip text

MagicDraw now displays a tooltip that shows supplementary information of what will happen whenever you drag any elements (Figure 107 on page 281, Figure 108 on page 282, Figure 110 on page 284).



Dragging a File to an Element

The improved drag-and-drop capability allows you to drag any files from your file system to any element in the browser or in a diagram. A hyperlink will be automatically created for the element to which the file is dragged, allowing you to open the file by double-clicking the element.

To drag a file on an element:

1. Select a file in your Explorer (Figure 106 on page 280).
2. Drag it to the element in the browser or in a diagram in MagicDraw (Figure 107 on page 281). A hyperlink to the file will be created.

Figure 107 on page 281 show how a hyperlink from the Products package to the Products_description.doc file is created.

5 WORKING WITH DIAGRAMS

Dragging, Copying, Cutting, and Pasting

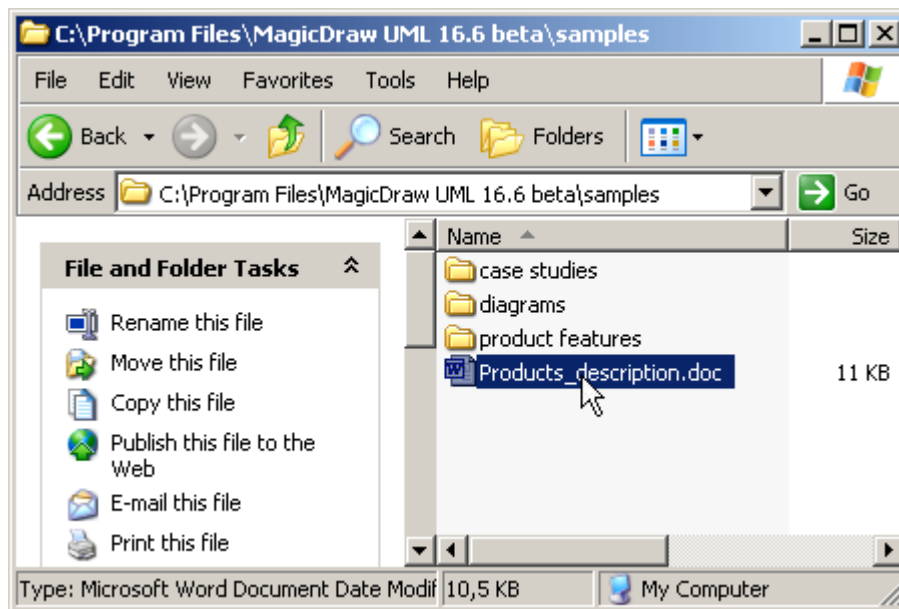


Figure 106 -- Selecting File in Your Explorer

5 WORKING WITH DIAGRAMS

Dragging, Copying, Cutting, and Pasting

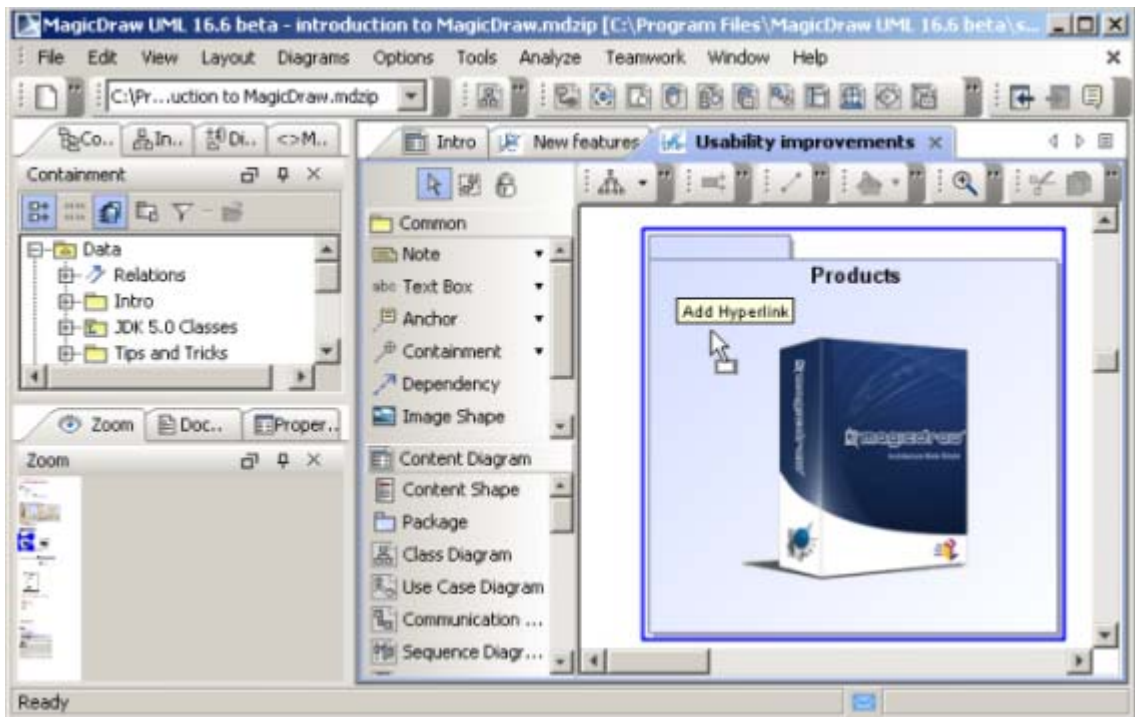


Figure 107 -- Dragging a File to the Element and Creating a Hyperlink



Drag and drop MagicDraw file on a diagram to open project

You can now drag MagicDraw project file from your file system and drop it on a diagram or any non-element. MagicDraw project will open (Figure 108 on page 282).

5 WORKING WITH DIAGRAMS

Dragging, Copying, Cutting, and Pasting

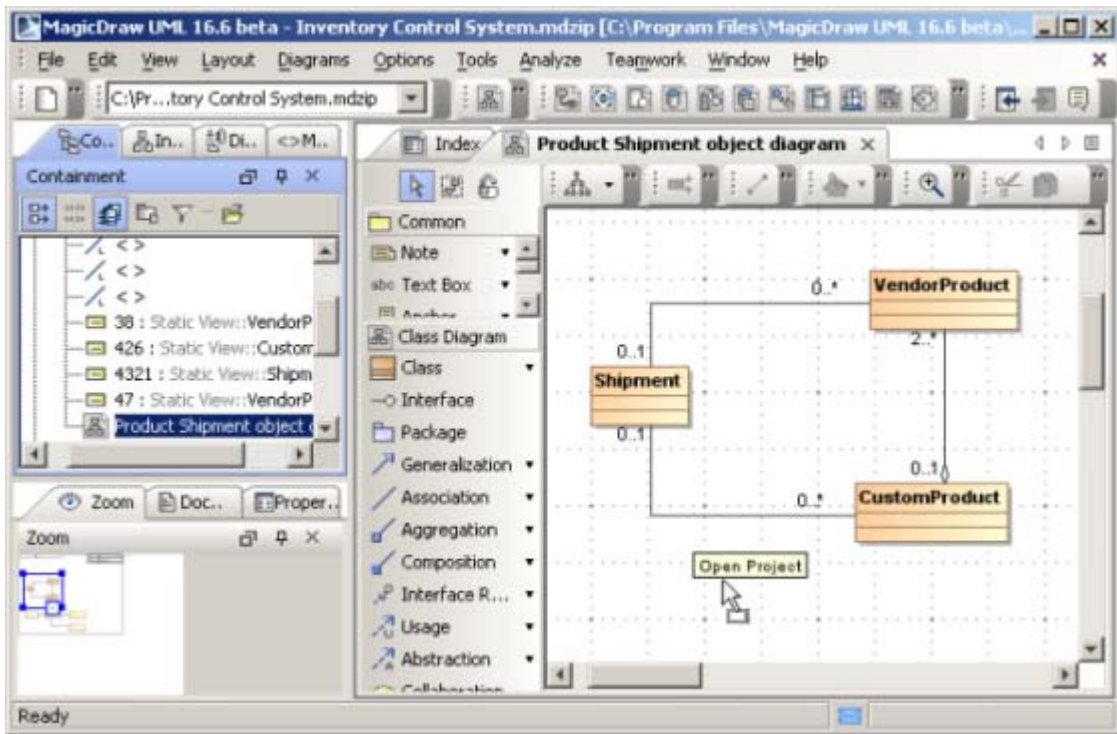


Figure 108 -- Dropping MagicDraw project file to diagram



Dragging an Image to an Element

You can now drag an image file from your file system to an element in the browser or in a diagram (Figure 109 on page 283). The image will be set as the value of the Image property of the element (Figure 110 on page 284).

The image will be set as a Stereotype icon if it is dragged to a Stereotype.

5 WORKING WITH DIAGRAMS

Dragging, Copying, Cutting, and Pasting



Figure 109 -- Image was Dragged and Dropped to the Class Shape

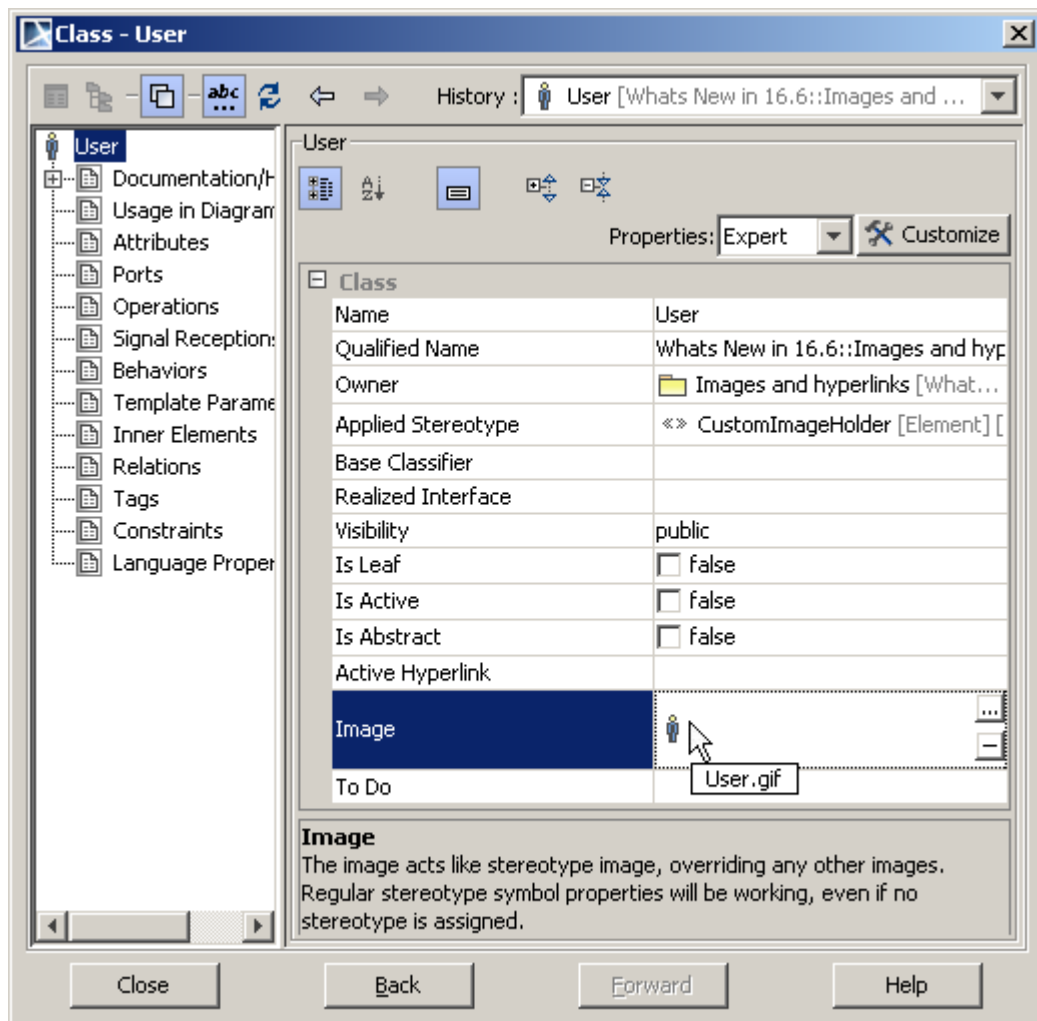


Figure 110 -- The Image Property in the Class Specification Dialog



Dragging Elements in the Specification Dialog

You can now drag any elements to any properties in the Specification dialog. For example, you can drag a Class element from the Containment tree to the **Type** property in the **Operation** Specification

dialog. The Specification dialog will then assign that Class element as the type of the Operation element. In this case, the step-by-step example is as follows:

1. Open the **Customer Class** specification dialog, the **Operations** branch, and select the *getProfile* Operation.
2. Select the *CustomerProfile* class in the Containment tree () and drag it to the **Type** property area in the open **Customer Class** specification dialog (Figure 111 on page 285). The *getProfile* operation type will be assigned to the *CustomerProfile* class.

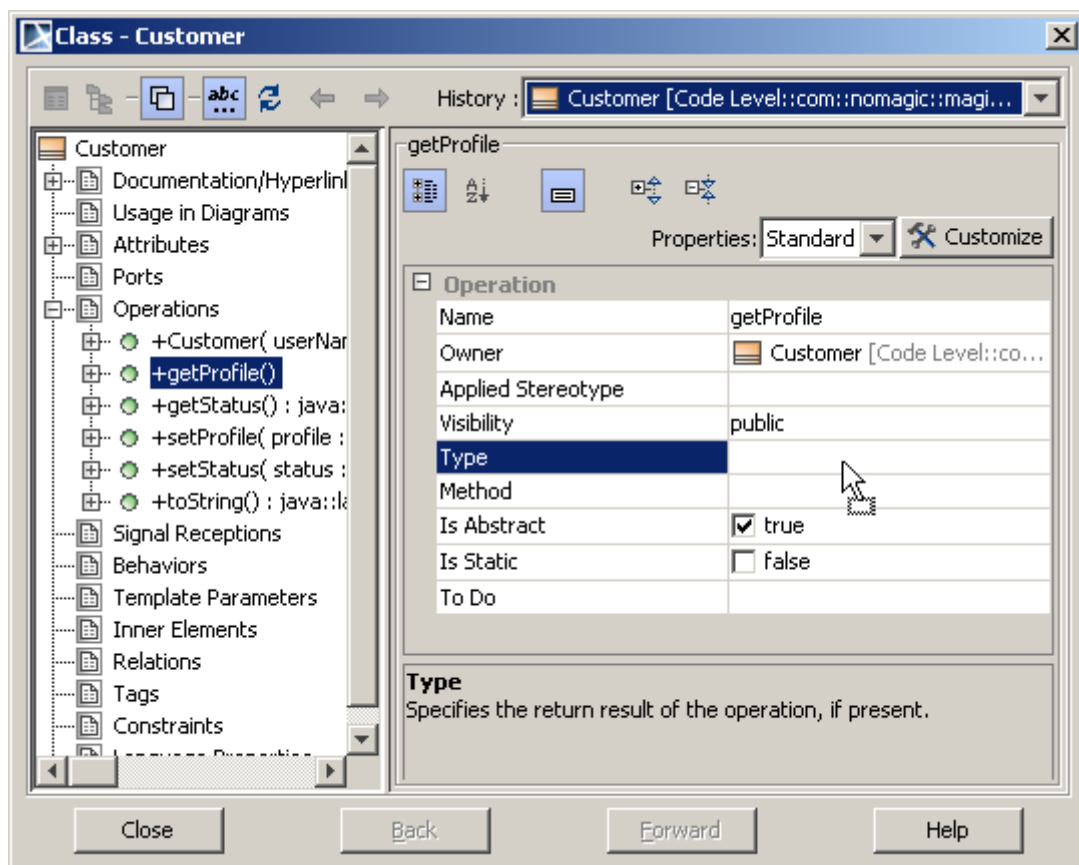


Figure 111 -- Dragging Class to the Property in the Specification Dialog

5 WORKING WITH DIAGRAMS

Dragging, Copying, Cutting, and Pasting



Dragging elements from the Specification dialog

You can now drag any elements from the Specification dialog to a diagram or to the browser. For example:

1. Open the Class specification dialog.
2. Select User Class, which is assigned as the Base Classifier (Figure 19).
3. Either (i) drag it to a diagram in the empty diagram pane to create a User Class symbol.
 - Or (ii) drag it to the existing shape to create a new Attribute with Type.

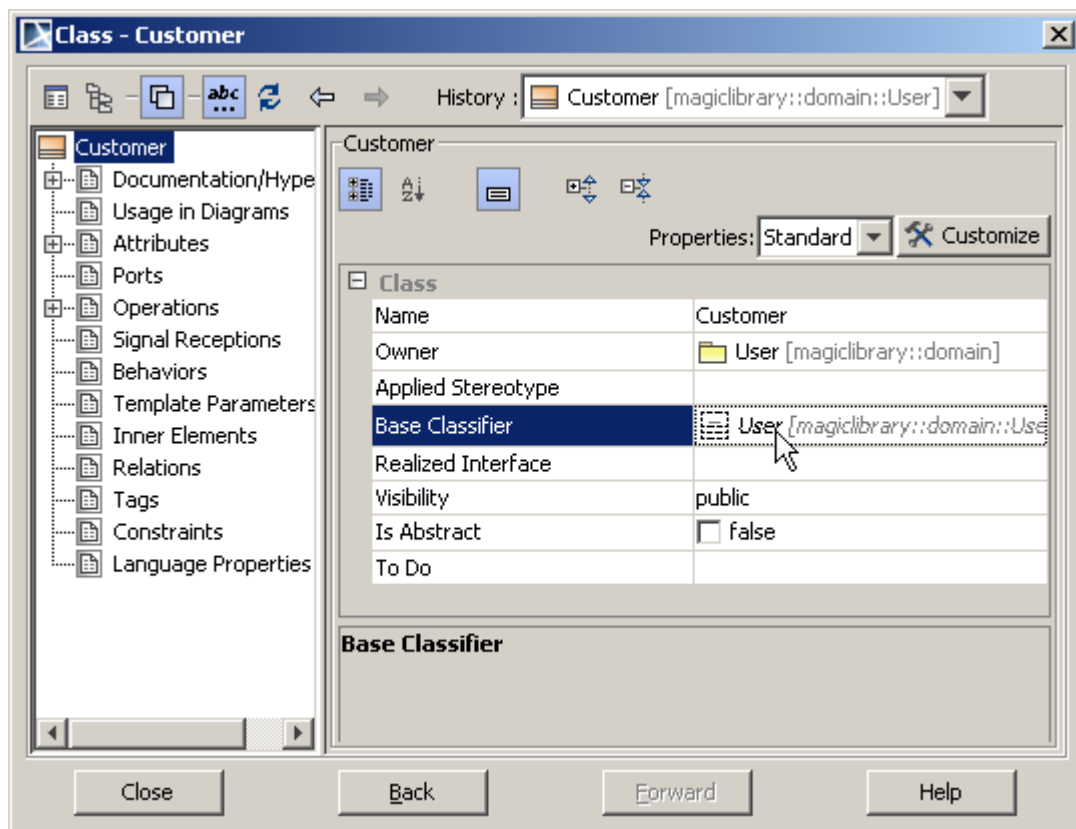


Figure 112 -- Dragging from the Customer Class Specification Dialog

5 WORKING WITH DIAGRAMS

Changing the diagram type



Drag and drop Stereotype

You can now drag Stereotype from Browser or Diagram on any other element to apply it.



Drag and drop in Sequence diagram

- You can now drag an Operation from the browser to a Message in a Sequence diagram. The message will become a Call Message with once the operation has been assigned.

NOTE

The Lifeline type must have/inherit this operation.

- Dragging a Signal to a Message in a Sequence diagram will convert the Message into a Send Signal Message and assign the Signal to the Message.



Drag-and-drop in State Machine Diagrams

You can now drag an Event element to a Transition element in a State Machine diagram. A Trigger with this Event will be created for that Transition element.



Drag-and-drop in Activity Diagrams

It is now possible to:

- Drag a Signal to an Activity diagram to create a Send Signal Action.
- Drag a Signal to a Send Signal Action to set or change the Signal.
- Drag an Event to an Activity diagram to create an Accept Event Action.
- Drag an Event to an Accept Event Action to set the Event.

Changing the diagram type

The diagram type may be changed to another type of diagram if both diagram types are compatible.

Note: diagram elements are not converted.

Changing the diagram type is usable:

- To migrate with existing project to the a diagram type which was not available till then. For example, to migrate Class diagram to the SysML Block Definition diagram.
- To migrate with existing project from the diagram type, which will be dropped from your project. For example, if user has decided to drop plug-in, and needs to convert plugin specific diagram to standard MagicDraw diagrams.

Diagram conversion scenarios:

- Any static diagram may be converted to another type of static diagram.
- Any dynamic diagram may be converted to another diagram, if both diagrams are based on the same diagram type and diagrams are compatible.

To change the diagram type:

1. Select one or more the same type diagrams in the Browser (Containment or Diagrams tree).
2. From the shortcut menu, choose command **Change Type To** and select desired diagram type from the list (see Figure 113 on page 289).

You can also convert multiple the same type diagrams – select them in the Browser and choose command **Change Type To**.

5 WORKING WITH DIAGRAMS

Changing the diagram type

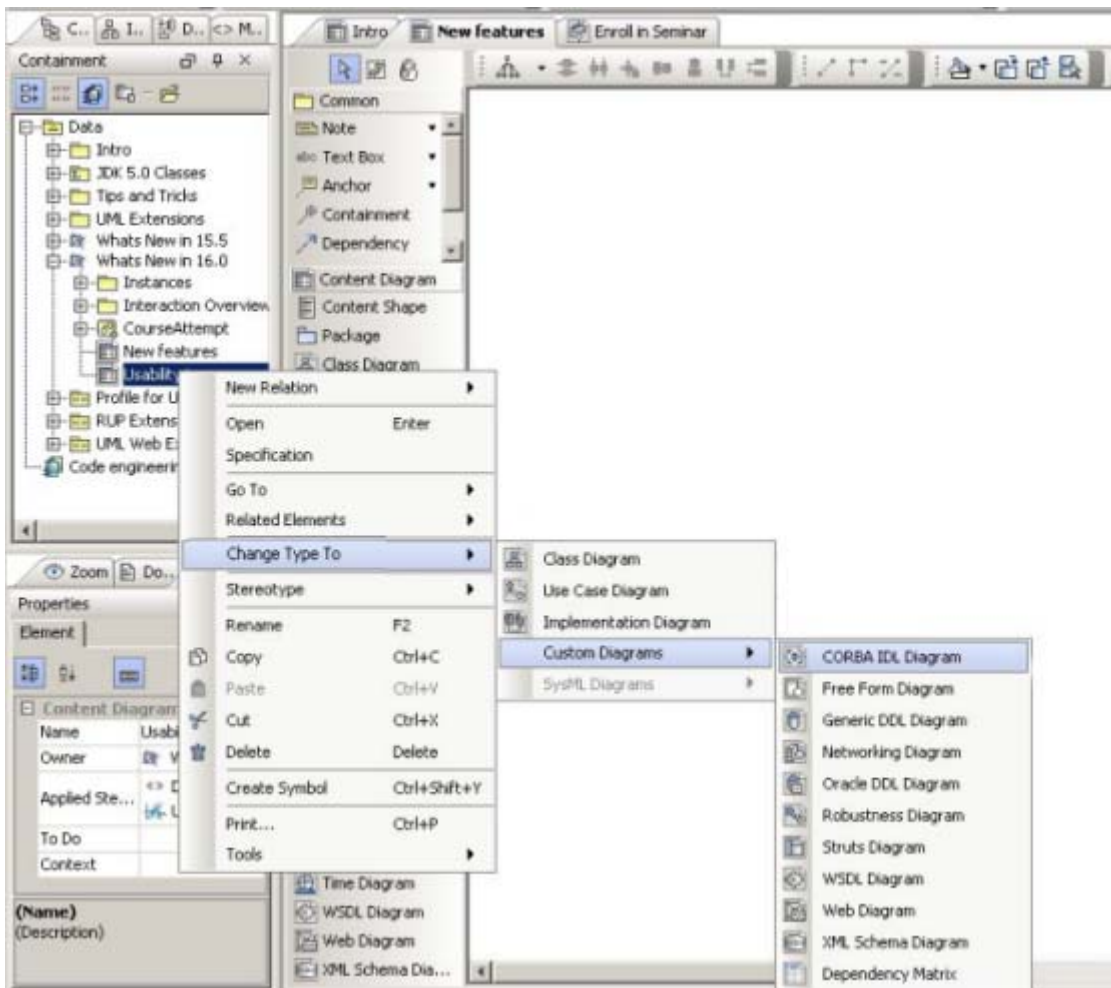



Figure 113 -- Changing the diagram type


Zooming

Zooming allows you to select a particular part of a diagram, zoom into it, and make changes while working with a finer level of detail. You can also gain an overview of a diagram by zooming out from it.


To fit the current diagram in the window

- From the **View** menu or from the diagram shortcut menu, select **Fit in Window** (shortcut keys CTRL+W.)
- In the Browser **Zoom** tab, click **Fit in Window** .

To zoom into the current diagram

- From the **View** menu or from the diagram shortcut menu, select **Zoom In** (shortcut keys CTRL+NumPad PLUS SIGN (+) or scroll.)
- Click the **Zoom In** toolbar button .


To zoom out from the current diagram

- From the **View** menu or from the diagram shortcut menu, select **Zoom Out** (shortcut keys CTRL+NumPad MINUS SIGN (-) or scroll.)
- Click the **Zoom Out** toolbar button .

TIP!

You can zoom in or zoom out using the CTRL+wheel keys.

To restore the diagram to the original size

- From the **View** menu or from the diagram shortcut menu, select **Zoom 1:1** (shortcut keys CTRL+NumPad SLASH MARK (/).)
- Click the **Zoom 1:1** toolbar button .

To view the selected shapes at maximum size

Select the shapes and then from either the **View** menu or the diagram shortcut menu, select **Zoom to Selection** (shortcut keys CTRL+NumPad ASTERICS MARK (*).)

To select the zoom settings

1. From the **Options** menu, select **Environment**. The **Environment Options** dialog box opens.
2. Open the **Diagram** pane and change the **Zoom Step Size** property. The maximum number is 1.0 (you may zoom a diagram twice.)

NOTE

You may also zoom in or out of the diagram using the zoom panel in the Browser window. For the detailed description, see “Documentation/Zoom Control/Properties” on page 126.

Using the Grid

The grid helps to arrange diagram symbols on the diagram pane. By default the grid is in the visible state.

To change the grid state (visible, not visible)

From either the **View** menu or from the diagram shortcut menu, select **Grid** and then select/clear the **Show Grid** check box.

To pull a path with the intersection of gridlines

From either the **View** menu or from the diagram shortcut menu, select **Grid** and then select/clear the **Snap Paths to Grid** check box.

To pull a shape with the intersection of gridlines

From either the **View** menu or from the diagram shortcut menu, select **Grid** and then select/clear the **Snap Shapes to Grid** check box.

To change the grid size

1. From either the **View** menu or from the diagram shortcut menu, select **Grid** and then select **Grid Size**.
2. The **Grid Size** dialog box opens.
3. Enter a grid size between 2 and 30 (default is 10).

4. Click **OK**.

To change the grid style

1. From the **Options** menu, select **Environment**. The **Environment Options** dialog box opens.
2. From the **Grid Style** drop-down list, select one of the following styles:
 - Dense
 - Sparse (default)

Layout

NOTE The diagram layout engine has nine layout options available in Standard, Professional, Architect, and Enterprise editions only.

With MagicDraw, it is easy to manage simple or complex diagrams using the automated layout features that optimize diagram layout for viewing.

Arrange your symbols on the Diagram pane using the **Layout** menu, or you can use the symbol shortcut menu when two or more symbols are selected. Since MagicDraw version 8.0, a new layout component has been applied with many more possibilities for arranging your models.

To resize the selected shape

Drag the corner of the shape to the desired size.

To automatically resize the selected shape to a preferred size

From the shape shortcut menu, select **Autosize**.

To alter the routing of the path line

Drag any point of the selected path in any direction.

The MagicDraw layout mechanism is built on various layout tools. All layout tools could be separated into 2 different groups: general layout tools and specific diagram layout tools. MagicDraw has only one specific diagram layout tool - the Class Diagram Layout Tool.

The General Layout Tools are:

- Orthogonal Layout Tool
- Hierarchic Layout Tool
- Tree Layout Tool
- Organic Layout Tool
- Circular Layout Tool
- Path Router

These layout tools are provided by Files layout tool component. You can arrange each diagram (except Sequence and Time Diagram) by using any of the 6 general layout tools. The Class diagram can also be arranged by using the Class Diagram Layout Tool.

Orthogonal Layout Tool

The Orthogonal Layout is well suited for medium sized sparse diagrams. It produces compact drawings with no shape overlaps, few crossings and few bends. All edges will be routed in an orthogonal style, i.e. only rectilinear style paths will be used.

Orthogonal layout options

Option	Values	Default Value	Description
Use Existing Drawing as Sketch	Boolean	False	The layout tool tries to "orthogonalize" the given sketch by interpreting it and without making too many modifications in respect to the original drawing.
Layout Only Top Level Symbols	Boolean	False	Keeps the relative position of symbols inside packages and performs the top level layout.

Option	Values	Default Value	Description
Group Layout Quality	0-1	1	Set the desired layout quality. Higher values result in less connection crossings and smaller layout area, but also increased computation time.
Orthogonal Grid	Integer	50	Defines the virtual grid spacing used by the layout tool. Each shape center point lies on a grid point.

Hierarchic Layout Tool

The Hierarchic layout can be used to highlight the main direction or flow within a diagram. Cyclic dependencies of shapes will be automatically detected and resolved. Shapes will be placed in layers, arranged by hierarchy. Additionally, the ordering of nodes within each layer is chosen in such a way that the number of path crossings is minimal.

Hierarchy layout options

Option	Values	Default Value	Description
Reverse Orientation in Activity Diagram	Boolean	True	If True, orientation is reversed in activity diagram.
Minimal Layer Distance	Integer	40	Determines the minimal distance between shapes that reside in adjacent layers.
Minimal Shape Distance	Integer	30	Determines the minimal distance between adjacent shapes that reside in the same layer.
Minimal Path Distance	Integer	30	Determines the distance between adjacent pairs of horizontal path segments and between horizontal path segments and shapes.

Option	Values	Default Value	Description
Minimal First Segment Length	Integer	10	Determines the minimal length of the first and last segments for orthogonal path routings, i.e. the length from the intersection points to the first or last bend point respectively.
Orientation	Top to Bottom, Bottom to Top, Left to Right, Right to Left	Top to Bottom	Determines main layout orientation.
Shape Placement	Linear Segments, Polyline, Simplex, Pendulum, Median Simplex, Tree	Simplex	<p>Linear Segments. Aligns shapes in such a way that path segments tend to have very few bends. It is a very good choice in combination with Path Routing set to Orthogonal. However, this greatly increases layout width.</p> <ul style="list-style-type: none"> • Polyline. Aligns shapes by slightly reducing the width without shape overlaps. Although, paths will have lots of bends. • Pendulum. A sound combination of Linear Segments and Polyline. • Simplex. Produces high quality drawings. Similar to Linear Segments, aligns shapes in such a way that path segments tend to have very few bends. Additionally, the resulting layout will be more balanced and more compact. • Median Simplex. Tends to produce more locally symmetric layouts for the sake of a few more bends. • Tree. Produces very nice layout, when the graph is a tree. If the graph is not a tree, the placement policy of Linear Segments will be used.

Option	Values	Default Value	Description
Path Routing	Oblique, Orthogonal	Orthogonal	<ul style="list-style-type: none"> • Oblique. Paths are routed to oblique style with a certain number of bends. • Orthogonal. Paths are routed to orthogonal style. Orthogonal path routing increases the height of the layout.
Randomization Rounds	Integer	40	Determines the number of rounds that are initialized using different randomized starting positions. Greater values can lead to fewer crossings and longer running times. Huge diagrams with lots of inherent crossings should be processed using smaller values.
Layout Only Top Level Symbols	Boolean	False	Keeps the relative position of symbols inside packages and performs the layout only on the top level.
Make Sub Trees	Boolean	True	Inheritance paths are joined into inheritance arcs.

Tree Layout Tool

The Tree layouter organizes diagram shapes into a tree structure. The Tree layout tool might be applied on shapes that have no undirected cyclic paths between them.

Table 3, Visible Tree layout options

Option	Values	Default Value	Description
Layout Style	Directed, Balloon, Horizontal-Vertical	Directed	<ul style="list-style-type: none"> • Directed. The tree is a hierarchy layout hierarchically with the root shape on the top. This is a good choice for directed trees with a unique root shape and a moderate number of shapes on a single hierarchy level. This layout style uses the current diagram layout as a sketch to determine the order of siblings at a common shape. • Balloon. The tree is routed in a radial style. This is a good choice for undirected, dense, or huge trees with a high number of shapes on a single hierarchy level. • Horizontal-Vertical. Children of a shape are either arranged on a horizontal or on a vertical line. Paths are routed orthogonally. This layout can be very compact if you choose the right alignment type for children of a node.
Make Sub Trees	Boolean	True	Inheritance paths are joined into inheritance arcs.
Directed			
Minimal Layer Distance	Integer	50	Determines the minimal distance between parent and child shapes.
Minimal Shape Distance	Integer	50	Determines the minimal distance between siblings of a shape.
Orientation	Top to Bottom, Bottom to Top, Left to Right, Left to Right	Top to bottom	Determines the main layout orientation. The layout tool tries to arrange shapes in such a way that all paths point in the main layout direction.

Option	Values	Default Value	Description
Port Style	Border Centered, Border Distributed	Border Centered	<p>Determines the way paths are attached to shapes.</p> <ul style="list-style-type: none"> • Border Centered. Paths are attached to the center of the border of corresponding shapes. • Border Distributed. Path attachment points are distributed along the border of corresponding shapes.
Orthogonal Path Routing	Boolean	True	If selected, all paths are routed orthogonally in a bus-like fashion. If not selected, paths are routed as straight line segments.
Balloon			
Root Shape Position	Directed Root, Center Root	Center Root	<ul style="list-style-type: none"> • Directed Root. Selects a shape with indegree zero if present. A good choice for directed root trees. • Center Root. Selects the root shape in such a way that the depth of the resulting tree is minimized.
Preferred Child Wedge	0-360	340	Determines the angular range of the sector that will be reserved for children of a shape. The remaining angular range will be automatically used to accommodate the edge that connects to the root node.
Preferred Root Wedge	0-360	360	Determines the angular range of the sector that will be reserved around the root shape to accommodate attached subtrees.
Minimal Path Length	Integer	50	Determines the minimal length of a path.
Compactness Factor	0.1-0.9	0.5	A smaller compactness factor will result in shorter paths and a more compact overall layout.
Horizontal-Vertical			

Option	Values	Default Value	Description
Horizontal Spacing	Integer	20	The minimal horizontal distance between adjacent shapes.
Vertical Spacing	integer	20	The minimal vertical distance between adjacent shapes.

Organic Layout Tool

The organic layout is well-suited for the visualization of highly connected backbone regions with attached peripheral ring or star structures. These structurally different regions of a network can be easily identified by looking at a drawing produced by this layout tool.

Organic layout options

Option	Values	Default Value	Description
Preferred Path Length	Integer	50	Specify the preferred length of all paths. The layout tool tries to arrange the shapes in such a way that paths have a determined path length.
Obey Shape Size	Boolean	True	If True, the distance between two shapes is calculated with respect to the size of the shape.
Gravity Factor	-0.2-2	2	Regulates the tendency of the shapes to be placed near the center of the diagram. The greater the factor is, the closer shapes are placed to the center of diagram. Negative values lead to huge layouts.
Path Attraction	0-2	2	Higher values make Layout tool obey the given preferred path length.
Shape Repulsion	0-2	0	Higher values result in greater shape distances.

Option	Values	Default Value	Description
Activate Tree Beautifier	Boolean	True	If True, optimizes tree-like substructures of the diagram. The optimization process might ignore some layout options.
Layout Only Top Level Symbols	Boolean	False	If True, the algorithm keeps the relative position of the symbols inside the packages and performs the layout only on the top level.
Package Shape Compactness	0-1	0.2	Control the compactness of the package shape. Larger values lead to more compact package shapes, but paths between packages may be longer and the shapes inside packages tend to get clutched together at the center of the package.

Circular Layout Tool

The Circular Layout produces arrangements that emphasize group and tree structures within a network. It partitions shapes into groups by analyzing the connectivity structure of the network. The detected groups are arranged on separate circles. The circles themselves are arranged in a radial tree layout fashion.

Circular Layouter options

Option	Values	Default Value	Description
Layout Style	Compact, Isolated, Single Cycle	Compact	<ul style="list-style-type: none"> • Compact. Each group will consist of shapes that are reachable by two disjoint paths. Shapes that belong to more than one group will be assigned exclusively to one group. • Isolated. Each group will consist of shapes that are reachable by two path disjoint paths. All shapes belonging to more than one group will be assigned to an isolated group. • Single Cycle. All shapes will be arranged on a single circle.
Minimal Shape Distance	Integer	100	Determines the minimal distance between borders of two adjacent shapes on a common circle. The smaller the distance, the more compact the resulting layout.
Auto Circle Radius	Boolean	True	If True, automatically determines the radius of each circle in the layout. An automatically chosen radius is usually the smallest possible radius that obeys Minimal Node Distance.
Fixed Circle Radius	Integer	200	If Auto Circle Radius is not set, this option determines the fixed radius for all circles in the resulting layout. Minimal Node Distance will be ignored in this case.
Preferred Child Wedge	0-360	340	Determines the angular range of the sector that will be reserved for children of a shape. The remaining angular range will be automatically used to accommodate the paths that connect to the root node.

Option	Values	Default Value	Description
Minimal Path Length	Integer	50	Determines the minimal length of a path that connects two shapes that lie on separate circles. The smaller the chosen value, the more compact the resulting layout.
Maximal Deviation Angle	10-360	100	The bigger the chosen value, the more compact the resulting layout. If the value is smaller than 90 degrees, the tree-edges might cross through the circularly arranged groups of shapes.
Compactness Factor	0.1-0.9	0.5	The smaller the compactness factor, the shorter paths and the more compact the overall layout.

Orthogonal Path Router

This layout routes paths using only vertical and horizontal line segments, while keeping the positions of shapes in the diagram fixed. The routed paths usually will not cross any shapes and will not overlap any other paths.

Orthogonal Path layout options

Option	Values	Default Value	Description
Minimal Distance	Integer	20	Specifies the minimal allowed distance between shapes and paths.
Use Existing Bends	Boolean	False	Specifies whether existing bends should be used as an initial solution for the new routing.
Route Only Necessary	Boolean	False	If True, only paths that violate the minimal distance criterion will be rerouted.

Organic Path Router

This layout routes paths using an oblique path style, while keeping fixed positions of shapes on a diagram. The routed paths usually will not cross any shapes and will not overlap any other paths.

Organic Path layout options

Option	Values	Default Value	Description
Minimum Path Distance	Integer	1	Determines the minimum distance between any two path segments.
Custom Minimum Distance to Nodes	Integer	10	Determines the distance between any path segment and any shape side. The path router strictly adheres to the set value. Router does not use this value by default due to increased calculation times.
Route on Grid	Boolean	True	If True, all paths are routed on grid lines from the predefined grid.
Space Driven Vs. Center Driven Search	0-1	1	Determines the ratio between two complementary weighting strategies when looking for a path, namely "center driven" and "space driven" weighting. Values closer to 0 lead to paths that are more distributed over the available space. Values closer to 1 give more emphasis to paths closer to a path center.
Local Crossing Minimization	Boolean	True	If False, the number of crossings seen at a shape side can increase considerably. Since this option has a positive effect on a diagram "readability," it is enabled by default.

Class Diagram Layout Tool

The Class diagram layout tool uses different layout algorithms to improve class diagram readability.

Class Diagram layout options

Option	Values	Default Value	Description
Minimal Layer Distance	Integer	50	Determines the minimal distance between parent and child shapes.
Minimal Shape Distance	Integer	50	Determines the minimal distance between the siblings of a shape.
Orientation	Top to Bottom, Bottom to Top, Left to Right, Right to Left	Top to bottom	Determines the main layout orientation. The layout tool tries to arrange shapes in such a way that all paths point in the main layout direction.
Compactness Factor	0-1	1	Adjusting this value can lead to a variety of differing layouts. For small values, the resulting layout will use more space and shapes tend to be far away from each other. Values around 0.5 lead to evenly distributed shapes, whereas values near 1.0 produce highly compact layouts.
Space Driven Vs. Center Driven Search	0-1	1	Determines the ratio between two complementary weighting strategies when looking for a path, namely "center driven" and "space driven" weighting. Values closer to 0 lead to paths that are more distributed over the available space. Values closer to 1 give more emphasis to paths closer to a path center.
Build Generalization Hierarchies	Boolean	True	Classes connected by generalization paths are organized into hierarchies.
Build Realization Hierarchies	Boolean	False	Classes connected by realization paths are organized into hierarchies.
Build Containment Hierarchies	Boolean	False	Classes connected by containment paths are organized into hierarchies.

Option	Values	Default Value	Description
Make Sub Trees	Boolean	True	If enabled, inheritance paths will be joined into inheritance arcs.

Activity Diagram Layout Tool

The Activity diagram layout tool uses different layout algorithms to improve activity diagram readability.

Activity Diagram layout options

Option	Values	Default Value	Description
Minimal Layer Distance	Integer	40	Determines the minimal distance between parent and child shapes.
Minimal Shape Distance	Integer	30	Determines the minimal distance between the siblings of a shape.
Minimal path distance	Integer	30	Determines the distance between adjacent pairs of horizontal path segments and between horizontal path segments and shapes.
Minimal first segment length	Integer	30	Determines the minimal length of the first and last segments for orthogonal path routings, i.e. the length from the intersection points to the first or last bend point respectively.
Orientation	Top to Bottom, Bottom to Top, Left to Right, Right to Left	Top to bottom	Determines the main layout orientation.

Option	Values	Default Value	Description
Shape Placement	Linear segments Polyline Simplex Pendulum Median simplex Tree	Simplex	<p>Linear segments - aligns shapes in such a way that path segments tend to have very few bends. It is a very good choice in combination with Path Routing set to Orthogonal. However, this greatly increases the width of the layout.</p> <p>Polyline - aligns shapes by slightly reducing the width without shape overlaps. Although paths will have lots of bends.</p> <p>Pendulum - a sound combination of Linear Segments and Polyline.</p> <p>Simplex - produces high quality drawings. Similar to Linear Segments, aligns shapes in such a way that path segments tend to have very few bends. Additionally, the resulting layout will be more balanced and more compact.</p> <p>Median Simplex - tends to produce more locally symmetric layouts for the sake of a few more bends.</p> <p>Tree - produces very nice layouts, when the graph is a tree. If the graph is not a tree, a placement policy of Linear Segments will be used.</p>
Path Routing	Oblique Orthogonal	Orthogonal	<p>Oblique - paths are routed to oblique style with a certain number of bends.</p> <p>Orthogonal - paths are routed to orthogonal style. Orthogonal path routing increases the height of the layout.</p>
Randomization Rounds	Integer	40	Determines the number of rounds that are initialized using different randomized starting positions. Greater values can lead to fewer crossings and longer running times. Huge diagrams with lots of inherent crossings should be processed using smaller values.
Layout only top level symbols	Boolean	False	Keeps the relative position of symbols inside packages and performs the top level layout.

For more information about the smart layout feature drawing diagram, see “Smart Activity Diagram layout” on page 696.

Business Process Diagram Layout tool

The Business process diagram layout uses different layout algorithms to improve business process diagram readability.

Business process diagram layout options

Option	Values	Default Value	Description
Minimal Layer Distance	Integer	40	Determines the minimal distance between shapes that reside in adjacent layers.
Minimal Shape Distance	Integer	30	Determines the minimal distance between borders of two adjacent shapes on a common circle. The smaller the distance, the more compact the resulting layout.
Minimal Path Distance	Integer	30	Determines the distance between adjacent pairs of horizontal path segments and between horizontal path segments and shapes.
Minimal first segment length	Integer	30	Determines the minimal length of the first and last segments for orthogonal path routings, i.e. the length from the intersection points to the first or last bend point respectively.
Orientation	Top to Bottom, Bottom to Top, Left to Right, Right to Left	Top to bottom	Determines the main layout orientation.

Option	Values	Default Value	Description
Shape Placement	Linear segments Polyline Simplex Pendulum Median simplex Tree	Tree	<p>Linear segments - aligns shapes in such a way that path segments tend to have very few bends. It is a very good choice in combination with Path Routing set to Orthogonal. However, this greatly increases the width of the layout.</p> <p>Polyline - aligns shapes by slightly reducing the width without shape overlaps. Although paths will have lots of bends.</p> <p>Pendulum - a sound combination of Linear Segments and Polyline.</p> <p>Simplex - produces high quality drawings. Similar to Linear Segments, aligns shapes in such a way that path segments tend to have very few bends. Additionally, the resulting layout will be more balanced and more compact.</p> <p>Median Simplex - tends to produce more locally symmetric layouts for the sake of a few more bends.</p> <p>Tree - produces very nice layouts, when the graph is a tree. If the graph is not a tree, a placement policy of Linear Segments will be used.</p>
Path Routing	Oblique Orthogonal	Orthogonal	<p>Oblique - paths are routed to oblique style with a certain number of bends.</p> <p>Orthogonal - paths are routed to orthogonal style. Orthogonal path routing increases the height of the layout.</p>
Randomization Rounds	Integer	40	Determines the number of rounds that are initialized using different randomized starting positions. Greater values can lead to fewer crossings and longer running times. Huge diagrams with lots of inherent crossings should be processed using smaller values.
Layout only top level symbols	Boolean	False	Keeps the relative position of symbols inside packages and performs the top level layout.

Quick Diagram Layout feature

You may use the Quick Layout feature when editing diagrams other than class or sequence. This feature applies recommended layout tool with default options on active diagram.

Label layout in the diagram

In MagicDraw 16.0 version there is improved label layout in the diagram. The following label positions are improved for paths, relationship ends, and shapes:

- Default label positions were reviewed and improved.
- Label positions after moving a path, shape, or related element.

Default label positions

Default label positions leaves after moving a path, shape, or related element if it is semantically logical decision. See an example below, there association multiplicities leaves at their default positions after class is moved.

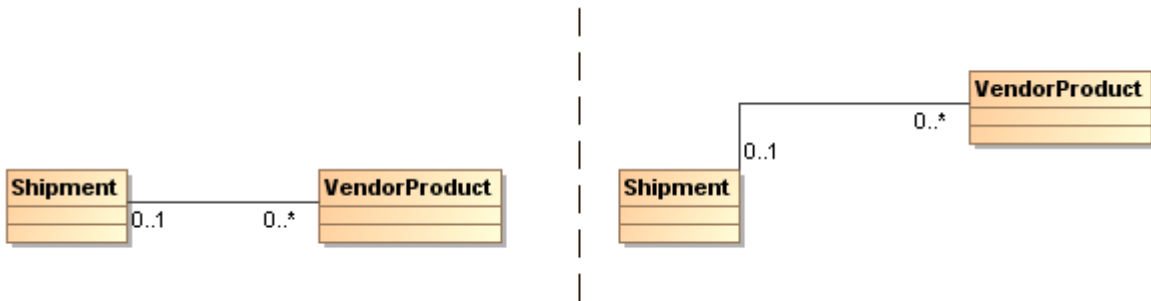


Figure 114 -- Default label position example

NOTE: If labels are at their default position, reset labels position functionality is disabled.

Labels positions after moving a path, shape or related element

After moving a path, shape or related element default label positions leaves if it is semantically logical decision.

For nicer representation of labels in diagram in the following cases labels positions are reseted to their default position automatically:

1. Symbol properties edit. When symbol properties edit causes label text box addition or removal from diagram pane labels positions are reseted.
2. Path, path end or port properties edit. When path, path end or port data edit causes label text box addition or removal from diagram pane labels positions are recalculated. See an example when qualifier is added in Figure 116 on page 310.
3. Path, shape or related element movement. See an example, when related element is moved Figure 117 on page 311.

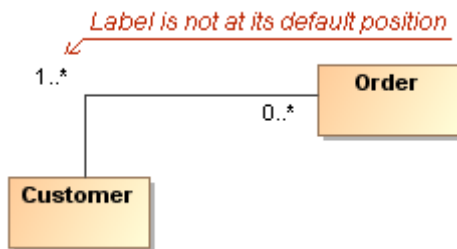


Figure 115 -- Label position before changes

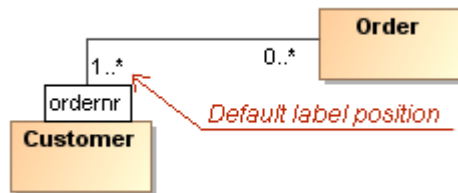


Figure 116 -- Label position is reseted to its default position after qualifier is added



Figure 117 -- Label position is reseted to its default position after Order class movement

Displaying label deviation from default position

While moving text box from default position, dotted line shows deviation from default position. This helps to see the current labels owner (See Figure 118 on page 311).



Figure 118 -- Dotted line, which shows deviation from the default position

Indicating label if it is not at its default position

If label is not at its default labels position, label right bottom corner is marked after label owner (path or shape) selection on diagram pane (see Figure 119 on page 312).



Figure 119 -- Marking the label when it is not at its default position

Showing Diagrams in Full Screen

If you want to see your diagrams in full screen and work exclusively from the diagram, use the (**show diagrams in full screen**) functionality. In full screen mode all necessary modeling commands will be visible, with option to hide, and the Browser will be in auto hide mode. You may also manage the MagicDraw interface components to be displayed or hidden.

To turn on diagram full screen mode

There are four ways to turn on the diagram full screen mode:

- Double click on the diagram tab with the diagram name at the top of diagram.
- From the diagram shortcut menu, select the **Show Diagrams in Full Screen** command.
- From the **View** menu, select the **Show Diagrams in Full Screen** command.
- Press the F11 key.

NOTE: You may change the **Show diagrams in full screen** shortcut key in the **Environment Options** dialog box, **Keyboard** pane. For more information, see "Assigning Shortcut Keys" on page 165.

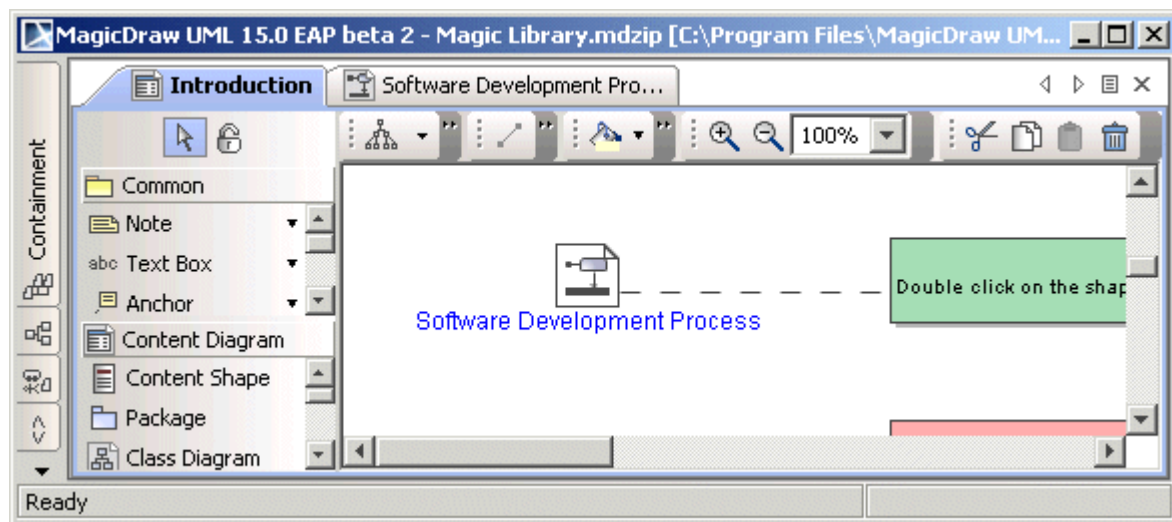


Figure 120 -- Diagram displayed in full screen mode

You may turn off the diagram full screen mode in the same way as turning it on.

Managing the MagicDraw interface components in the diagram full screen mode:

1. *Browser windows* are in auto hide mode when the diagram full screen mode is turned on. Located at the left side of the window are tabs of browser windows. To display the browser, move the mouse cursor over the browser window tab (for example, on the Containment tab) and the browser window will open. For more information about working with windows in auto hide mode, see "Using the Browser" on page 105.
2. *The main toolbar* is hidden when the diagram full screen mode is turned on. To display the main toolbar, clear the **Hide toolbars in full screen mode** check box in the **Environment Options** dialog box, **General** pane, **General** group.
3. *The diagram toolbar* is displayed when the diagram full screen mode is turned on. Right click the diagram toolbar to manage it.

NOTE

Showing the diagram in full screen mode is available only in Single Window (JIDE) interface style.

Floating Diagram Window

Floating diagram windows can be enabled by clicking the **Floating** command from the Diagram tab shortcut menu (Figure 121 on page 314).

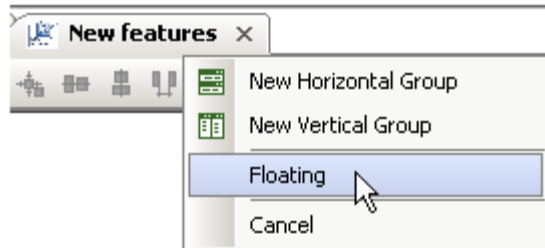


Figure 121 -- The Floating Command on the Diagram tab shortcut menu

Saving as an Image

Saving a diagram and selected symbols as an image

Diagrams that were created in the model can be saved as an image in the following formats:

- Enhanced Metafile Format (*.emf) - supports language specific symbols.
- Encapsulated PostScript (*.eps)
- Joint Photographic Experts Group (*.jpg, *.jpeg)
- Portable Network Graphics (*.png)
- Scalable Vector Graphics (*.svg)
- Tagged Image File Format (*.tif, *.tiff)
- Windows Metafile Format (*.wmf)

MagicDraw version 15.0 and above allows exporting of a created diagram to Tagged Image File Format (TIFF) and choice of desired color space and compression. TIFF images can be edited and

resaved without suffering a compression loss and it is a flexible and adaptable file format for high color depth images. TIFF format is superior to JPG format.

To save the current diagram or the selected elements within the diagram as an image

1. From the **File** menu, select **Save As Image**. The **Save As Image** dialog box opens.
2. Select the **Active Diagram** or the **Selected Symbols** option button.
3. Select the image format (*.emf, *.eps, *.jpg, *.png, *.svg, *.tif, *.tiff, *.wmf), file name, and the location directory.

To save the selected diagrams of your project as images

1. Select the **Save As Images** command from the **File** menu. The **Save As Image** dialog box opens.
2. Check the **Selected diagrams** radio button, and select the diagrams you want to save as images from the **Not empty diagrams** list.
3. In the **Working Directory** field, type in the name of the destination directory or click the '...' button to browse to the directory list.

5 WORKING WITH DIAGRAMS

Saving as an Image

4. Select the graphical file format in the **Image Format** drop down list (EMF, EPS, JPG, PNG, SVG, TIF, or WMF) and click **Save**.

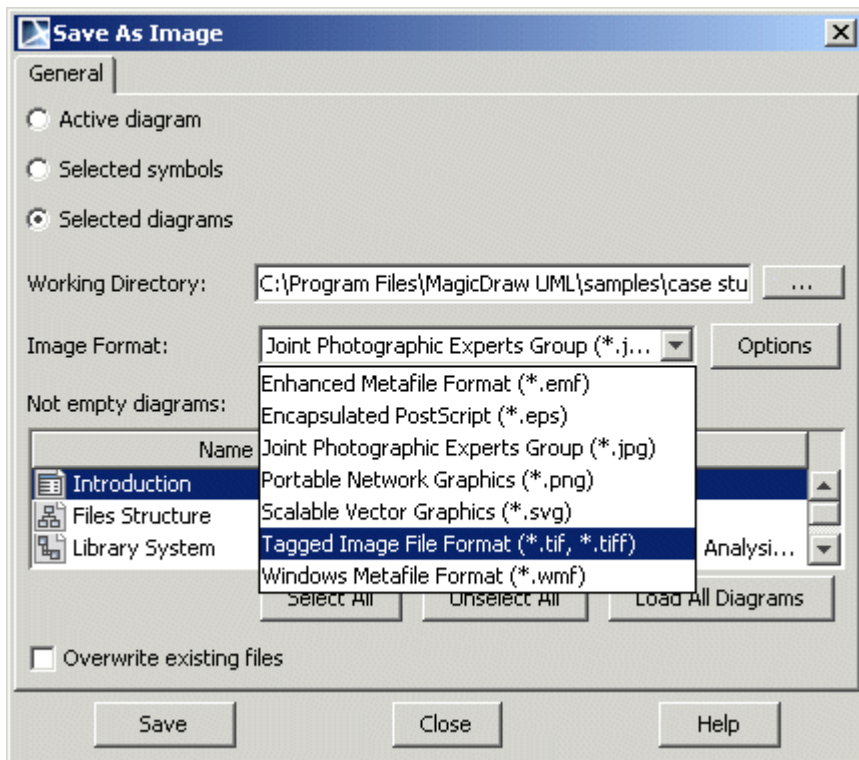


Figure 122 -- Save As Image dialog box

The filename of the saved diagram will be the same as the name of that diagram.

The **Not empty diagrams** list contains all exportable diagrams that contain UML elements. Select the diagrams you wish to export.

To make multiple selections

Press the CTRL key and click the diagrams you wish to export.

To select or clear all diagrams

Click the **Select All** button (press CTRL+A keys) or **Unselect All** button.

To display the list of all diagrams that are available in the project

Click the **Load All Diagrams** button in the **Save As Image** dialog box.

Setting image saving options

MagicDraw version 15.0 and above allows changing image size, resolution (DPI), and specifying other image properties specific to the selected image format.

To specify image options:

1. Select the **Save As Images** command from the **File** menu. The **Save As Image** dialog box opens.
2. Select the graphical file format in the **Image Format** drop down list.
3. Click the **Options** button near the **Image Format** drop down list. The **Image Export Options** dialog box opens.

Modify image export properties, which are described in the table below:

Property name	Description	Formats
Save diagram background in image	Saves the diagram with background. By default, the diagram background is white after saving as an image.	This property is included in all format options.
Image resolution (DPI)	This property is the DPI property value with numeric value range from 1 to 4800. Default value is 72.	Property is not included in <i>SVG</i> and <i>WMF</i> format options.
Exported image size [%]	Define image size in percent. Default value is 100%. For example, if 200% is defined, then the view is enlarged (zoomed) before generating the image. Raster image will not lose its quality as additional pixels are introduced.	Property is not included in <i>WMF</i> format options list.

Other image export options:

- **JPEG Compression Quality** property is included in *JPEG* format options list.
- **Use SVG <tag> for text output** property is included in *SVG* format options list.
- **Compression, Color space** properties are included only in *TIFF* format options list.

You may also define image saving options in the **Environment Options** dialog box. For more information see “Setting Environment Options” on page 129.


Printing

In MagicDraw you can print an active diagram, multiple diagrams, or selected model elements. All printing menu commands are found in the **File** menu. You can also use the toolbar buttons or the shortcut keys.

Before printing, use the **Print** dialog box to set your printing options.

NOTE	If the size of the text is too small, it may not be visible on the printed page.
-------------	--

To open the **Print** dialog box

- From the **File** menu, select **Print**.
- In the **Print Preview** screen, click the  button.

The **Print** dialog box contains the following tabs: **Print Range**, **Print Options**, and **Print Header/Footer**. Descriptions for these tabs appear in the following sections.

Print Range tab

In the **Print Range** tab, select the item you want to print.

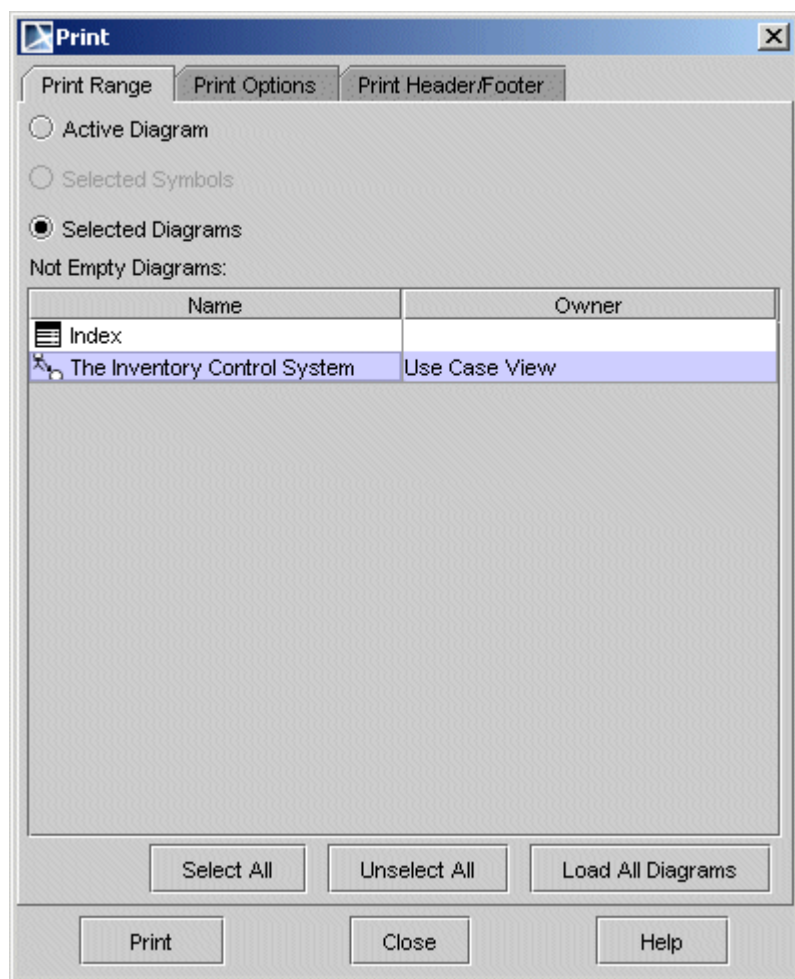


Figure 123 -- Print dialog box. Print Range tab

Element name	Function
Active Diagram	Print the currently open diagram.

Element name	Function
Selected Symbols	Print symbols you select on the diagram. The desired symbols should be selected to activate this option button.
Selected Diagrams	From the Not Empty Diagrams list, select the diagrams you want to print.
Name	Show available diagrams in the project. To select the diagram for printing, click the name of the diagram in the list. The selected diagrams are highlighted. Press CTRL or SHIFT to select more than one diagram.
Owner	The name of the model element that owns the particular diagram.
Select All	Select all diagrams in the list for printing.
Unselect All	Remove the selection.
Load All Diagrams	Load all diagrams belonging to a project. By default only open diagrams are shown in the Print dialog box.
Print	Print the selected diagram(s).
Close	Close the dialog box.
Help	Display MagicDraw UML Help.

Print Options Tab

Click the **Print Options** tab to customize the printing jobs.

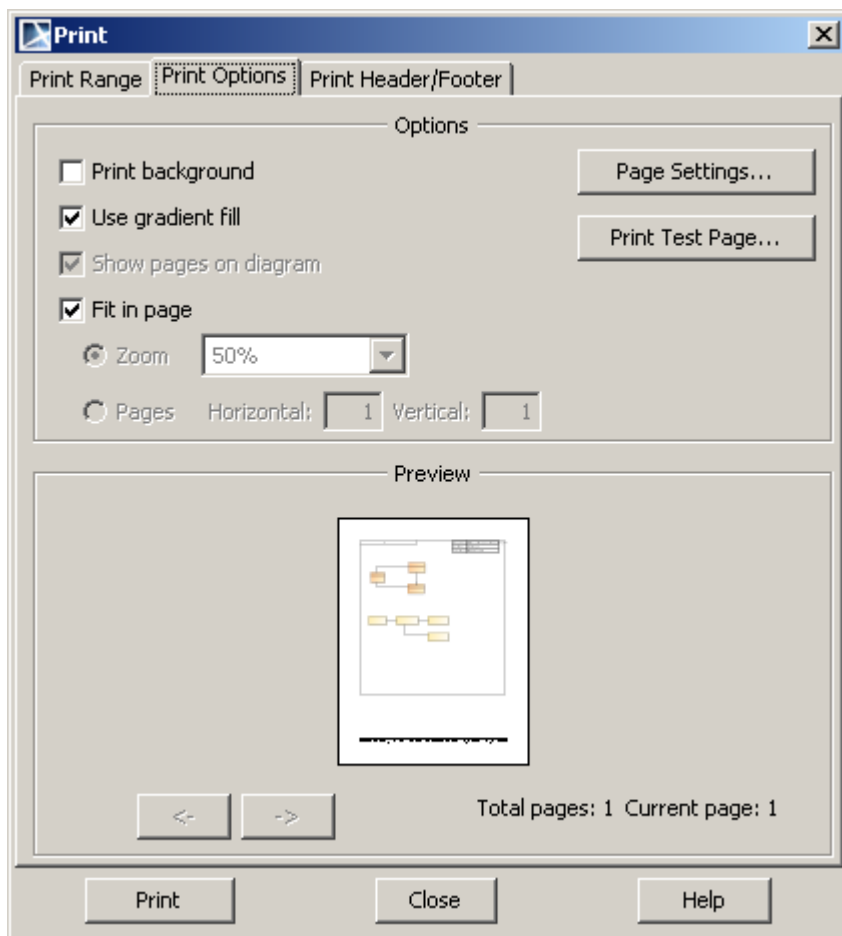


Figure 124 -- Print dialog box. Print Options tab

Element name	Function
Print Background	Print the background color of your diagrams.
Use gradient fill	Select this option to enable diagram symbols gradient fill in printing.

Element name	Function
Show Pages on Diagram	Show the page boundary on the diagram pane. NOTE You will not see any boundary if the Fit in Page check box is selected.
Fit in Page	The printed diagram fits in one page. If the Fit in Page check box is cleared and the Show Pages in Diagram check box is selected, the gridlines of pages are shown on the diagram pane.
Zoom	Zoom the selected diagram to the size you want for printing. NOTE You are not allowed to zoom a diagram if the Fit in Page check box is selected.
Pages	Set the number of pages on which you want to print the diagram. <ul style="list-style-type: none"> • Vertical. The number of vertical pages on which the diagram will be displayed. • Horizontal. The number of horizontal pages on which the diagram will be displayed.
Page Settings	The Page Setup dialog box opens.
Print Test Page	Print the test page. Set print options in the Print Options dialog box.
Preview	Preview the diagram appearance before printing.
<	Preview the previous page.
>	Preview the next page.

Print Header/Footer Tab

Click the **Print Header/Footer** tab to customize the header and footer of the printed pages.

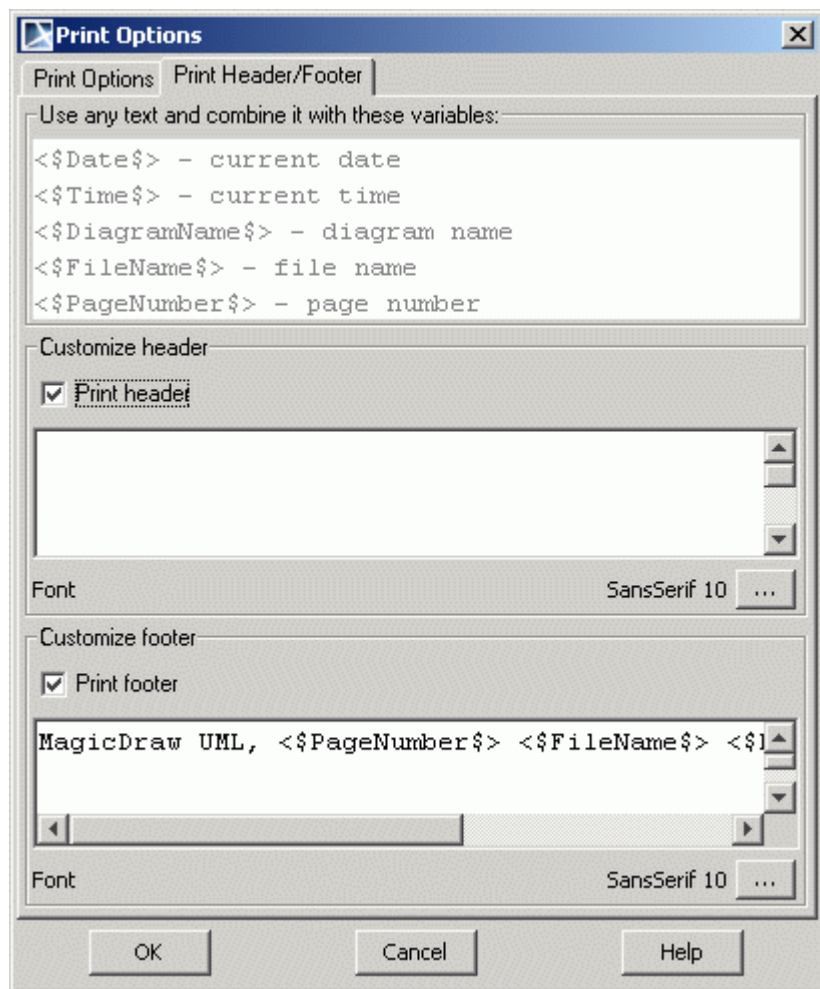


Figure 125 -- Print dialog box. Print Header/Footer tab

- **Use any text and combine it with these variables** box - use this box to indicate which fields you want to include in the header/footer.
- **Customize header** group box - prints the header. Select the **Print Header** check box and type the text you wish to be printed. Use the "... " button to select the desired font.

- **Customize footer** group box - prints the footer. Select the **Print Footer** check box and type or change the text you wish to print. Use the "..." button to select the desired font. By default <\$PageNumber\$> <\$FileName\$> <\$DiagramName\$> <\$Date\$> <\$Time\$> is printed.

6 WORKING WITH MODEL ELEMENTS

Specifying Model Elements

[View Online Demo](#)

Specification Windows

You can define all model elements in the corresponding Specification dialog boxes.

The MagicDraw shortcut menus, toolbars, and browser help ease the task of editing model elements.

IMPORTANT

Beginning with version 8.0, MagicDraw enables the editing of model elements and symbol properties directly from the Browser, located in the Quick Properties tab. For more information, see “Properties tab” on page 128.

Specification dialog boxes

The Specification dialog boxes are used to define UML model elements such as class, package, activity, and others. Every Specification dialog box has a different structure based on the element characteristics.

Model elements that may participate in the relationships contain the **Relations** tab. All model element specification dialog boxes have **Template Parameters**, **Tags**, **Constraints**, and **Documentation/Hyperlinks** tabs. Descriptions of these tabs are presented in the following sections.

To open the corresponding Specification dialog box

-
- Select **Specification** from the selected symbol shortcut menu.
 - Double-click a symbol on the Diagram pane or Browser.
 - Select a symbol on the Diagram pane and press the ENTER key.
 - The element Specification dialog box opens when you add one model element to another model element from another Specification dialog box. The second Specification dialog box opens on top of the first. Use the **Back to** or **Forward to** arrow buttons for switching between windows.

General tab

Name text box

Name	
------	--

Type or view the model element name. If you enter the name of an existing model element, an error message opens.

For some model elements (attribute, operation, and so forth), the default name *Untitled 1* is set. You can change this name to a preferred name.

Is Active, or Is Abstract check boxes

Is Active	<input type="checkbox"/> false
Is Abstract	<input type="checkbox"/> false

When one of these check boxes is selected, the model element is correspondingly set as an active or abstract generalizable model element.

A generalizable element is a model element that may participate in a generalization relationship.

Name	Function
Is Abstract	Specifies whether the generalizable element may or may not have a direct instance. True indicates that an instance of the generalizable element must be an instance of a child of the generalizable element. False indicates that there may be an instance of the generalizable element that is not an instance of a child. An abstract generalizable element is not instantiable since it does not contain all necessary information.

Applied Stereotype

Applied Stereotype	
--------------------	--

Click the "... " button to open the list of all available applied stereotypes, select the check box for the chosen stereotype and click **Apply**.

Visibility

Visibility	public
------------	--------

To define an element access level, use the drop down list to set its visibility. There are four levels of access:

- **Public.** The element can be accessed by any outside object.
- **Package.** The element can be accessed by any classifier declared in the same package (or a nested subpackage, to any level).
- **Private.** The element can be accessed only from inside the current class.
- **Protected.** The element can be accessed from inside the current class and classes derived from that class.

ToDo

To Do	
-------	--

Type or view information about an element. The **To Do** property is used for keeping special information, exclusive cases, or additional records.

Image

Click the “...” button to assign the image to the element. Assigned image can be displayed on the shape or instead of the shape.

For more information, about changing the image display mode, see “Displaying icon or image on the symbol or instead the symbol” on page 344.

Documentation/Hyperlinks tab

Use the **Documentation/Hyperlinks** tab to add comments to the selected element and to assign hyperlinks. The hyperlink can direct the user to a model element, web page, or a file.

6 WORKING WITH MODEL ELEMENTS

Specifying Model Elements

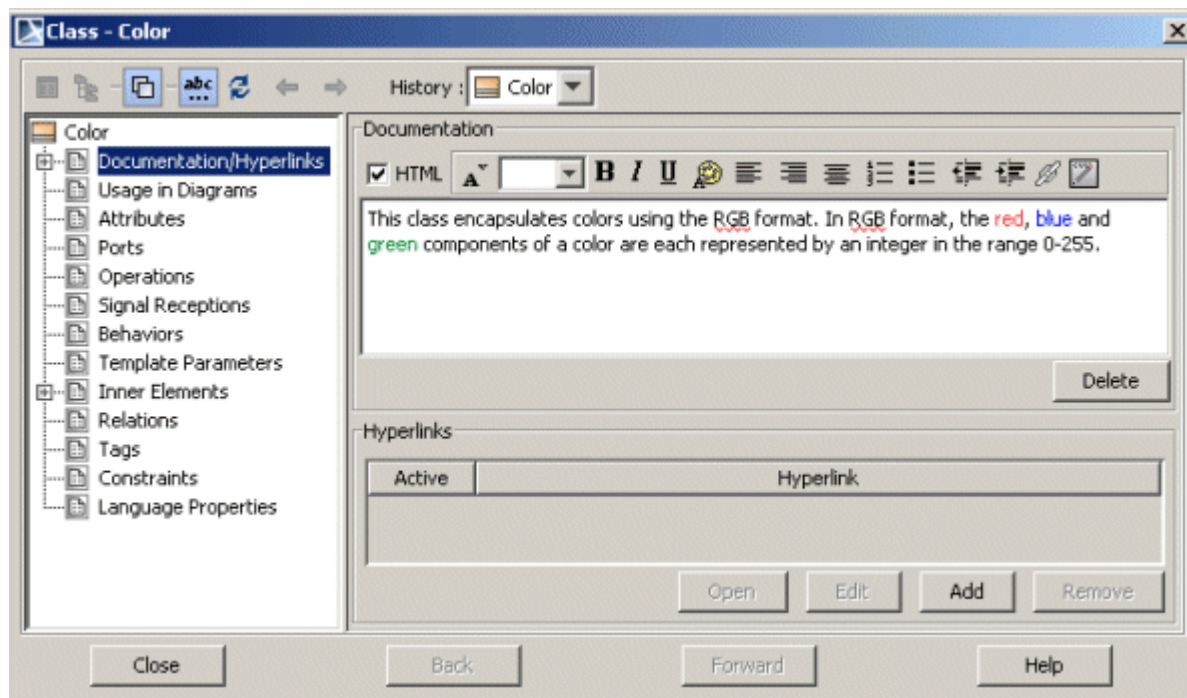


Figure 126 -- Specification dialog box. Documentation/Hyperlinks tab

Writing HTML documentation

To write documentation in HTML format, simply select the HTML check box to display a menu with the available text formatting options.

For more information about HTML editor toolbar, see "HTML editor toolbar" on page 381.

Adding Hyperlinks

In the **Hyperlinks** group, manage the hyperlinks you want to add to the model element

Active	If selected, the hyperlink is activated and will work when double-clicking the model element.
---------------	---

Hyperlink	Display information about the hyperlink: diagram or element name, file path, or URL name.
Open	Opens the previously assigned hyperlink.
Edit	The Insert Hyperlink dialog box opens. Edit the selected hyperlink.
Add	The Insert Hyperlink dialog box opens. Select the hyperlink you want to add to the model element.
Remove	Remove the selected hyperlink from the model element.

Attributes tab

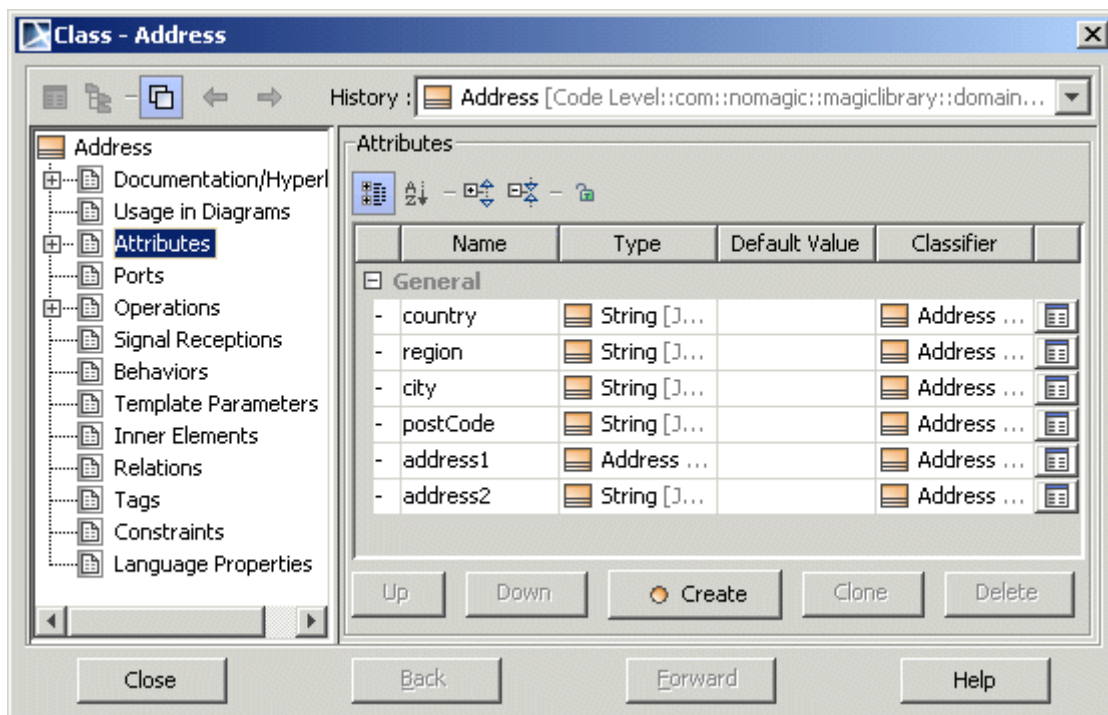



Figure 127 -- Attributes tab

The **Attributes** tab contains the model element attributes list and buttons for editing the attributes list.

Name	Attribute name.
-------------	-----------------

6 WORKING WITH MODEL ELEMENTS

Specifying Model Elements

Type	Attribute type. It can be a primitive type or another class.
Default Value	Attribute default value.
Classifier	Class name that contains the current attribute.
Up	Move item to upper position in the list.
Down	Move item to lower position in the list.
Create	Add a new attribute to the class. The Attribute Specification dialog box opens.
Clone	Enabled when the element is selected in the list. A new element will be created. The new element derives all properties from the cloned element. The name will be changed to "<element_name><number>".
Delete	Remove the selected attribute from the class.
	Click this button to open the Attribute Specification dialog box.

Usage in Diagrams tab

For more information about searching for symbol usage in diagrams from the **Usage In Diagrams** branch, see "Searching for symbol usage in diagrams from the element specification dialog box" on page 645.

Operations tab

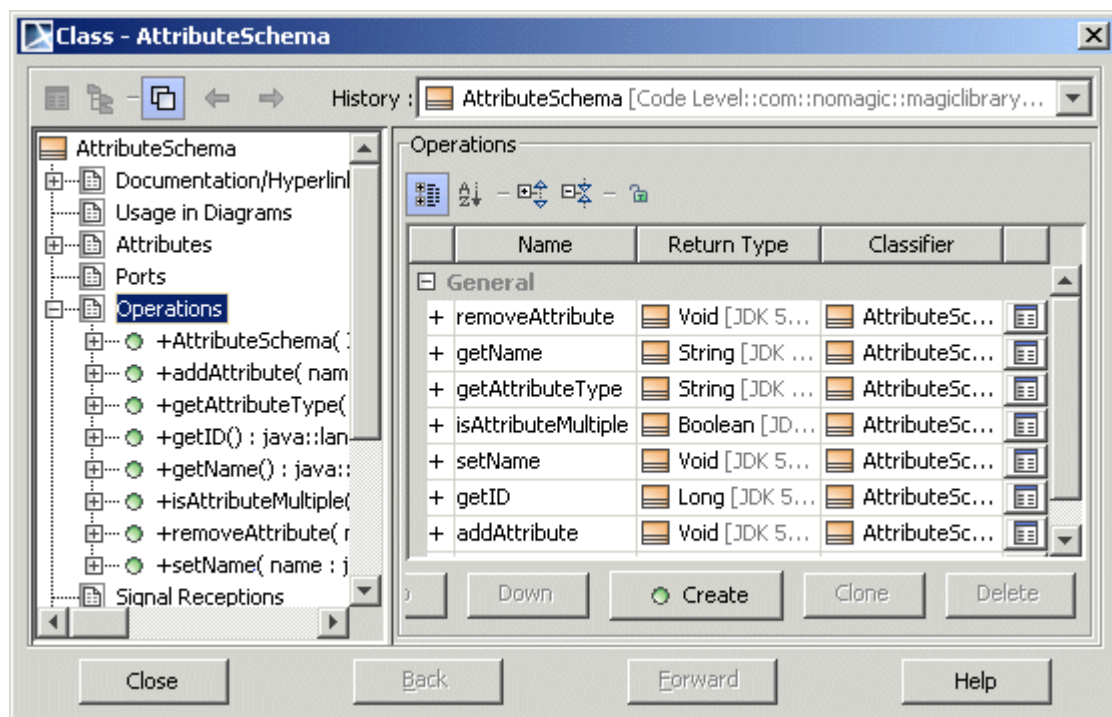



Figure 128 -- Operations tab

The **Operation** tab contains the model element operations list and buttons for managing this list.

Name	Operation name.
Return type	Operation return type.
Classifier	The name of the classifier containing the current operation.
Up	Move item to upper position in the list.
Down	Move item to lower position in the list.
Create	Add a new operation to the model element. The Operation Specification dialog box opens.
Clone	Enabled when the element is selected in the list. A new element will be created. The new element derives all properties from cloned element. The name will be changed to "<element_name><number>".

Delete	Remove the selected operation from the model element.
	Click this button to open the Operation Specification dialog box.

Template Parameters tab

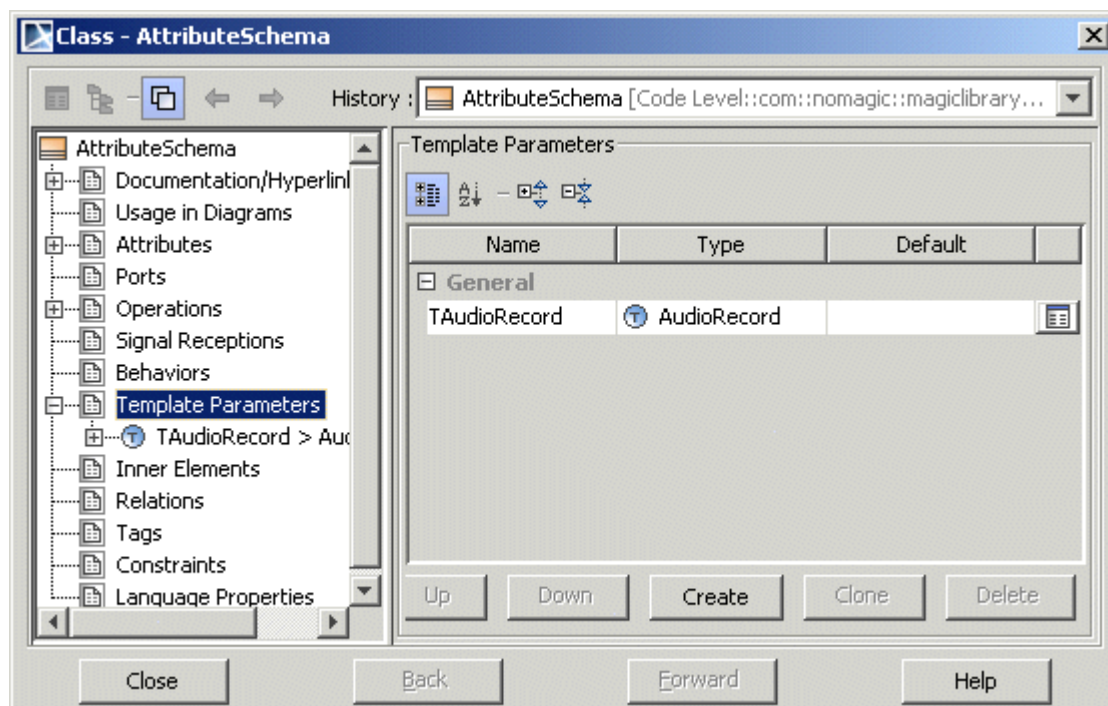



Figure 129 -- Operations tab

The **Template Parameter** tab contains the model element template parameters list and buttons for managing this list.

Name	The name of the template parameter.
Type	Type of template parameter: classifier or data type.
Default	The Select Element dialog box opens. Here you can assign the element as the default element for the template parameter.
Up	Move the item to the upper position.

6 WORKING WITH MODEL ELEMENTS

Specifying Model Elements

Down	Move the item to the lower position.
Create	Create a new template parameter.
Clone	Enabled when the element is selected in the list. A new element will be created. The new element derives all properties from the cloned element. The name will be changed to "<element_name><number>".
Delete	Remove the template parameter from the class.
	Click this button to open the Template Parameter Specification dialog box.

Relations tab

The **Relations** tab contains the list of relationships in which the appropriate model element participates.

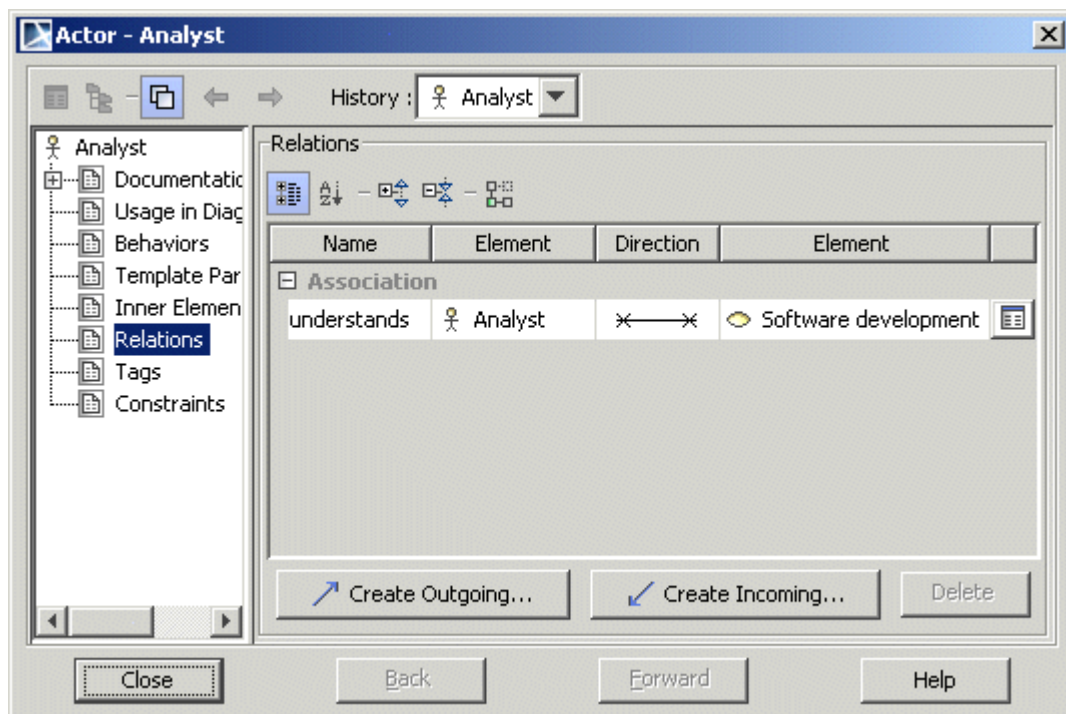


Figure 130 -- Specification dialog box. Relations tab

Element name	Function
Name	Name of the relationship (optional).
Element	One of the relationship endings.
Direction	Shows a relationship's direction, helps to specify source and target.
Element	Another relationship ending.

6 WORKING WITH MODEL ELEMENTS

Specifying Model Elements



After clicking this button, the relationship **Specification** dialog box opens.

Delete

Removes the selected relationship from the list.

Tags tab

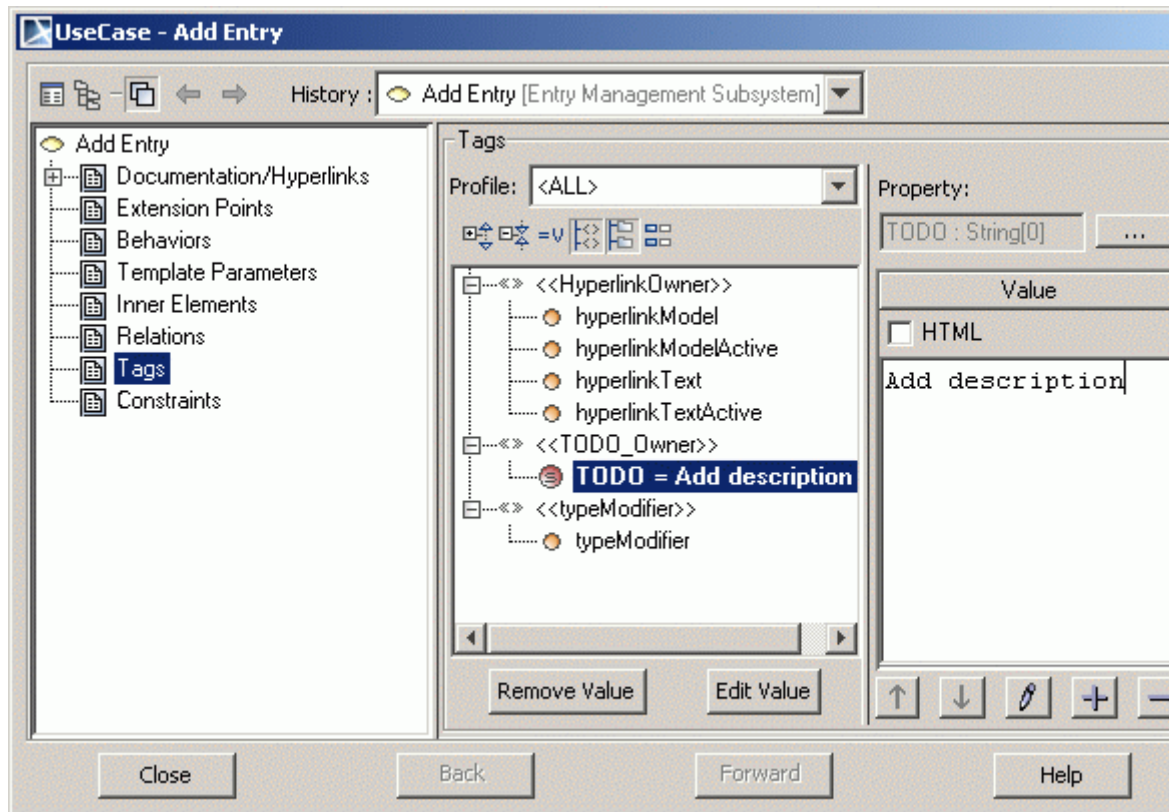








Figure 131 -- Specification dialog box. Tagged Values tab

Element name	Function
Profile	Lists the profiles available for the current project.

6 WORKING WITH MODEL ELEMENTS

Specifying Model Elements

Element name	Function
 Expand All Tree Branches	If tag definitions are grouped and those groups are collapsed, expands the groups.
 Collapse All Tree Branches	If tag definitions are grouped and those groups are expanded, collapses the groups.
 Show Tag Definitions with Values	Only displays in the list those tag definitions that have values.
 Show Tag Definitions Groups	If tag definitions are grouped into specific “packages”, shows those “packages” on the list by grouping tag definitions.
 Group By Stereotype	Sorts tag definitions by the assigned stereotypes.
 Show Tag Definitions Types	Displays types of tag definitions in the list.
Create Value	The list of tag definitions with values that are selected for the current model element.
Create Value	Creates a value for the selected tag definition. The right pane of the dialog box is activated. Select or enter the value. All data types and types of metamodel can be types of values.
Remove Value (available only when the tag definition has a value)	Removes the value(s) from the selected tag definition.
Edit Value	The Tagged Value Specification dialog box opens, allowing you to edit or extend the selected tagged value.
Edit Tag Definitions	The Profiles dialog box opens. Edit, add, or remove tag definitions for the current model element.

6 WORKING WITH MODEL ELEMENTS

Specifying Model Elements

Element name	Function
Right pane of the dialog box	
Tag Definition “...”	Click the “...” button and edit the selected tag definition in the Tag Definition Specification dialog box.
HTML	Set the tagged value text as HTML.
Value (if the value is added)	Type or select the value.
Edit	Edit the selected value.
Add	Add a new value.
Remove	Remove the selected value.

For more information about how to create a new tagged value, see “To create a new tag definition” on page 812.

Constraints tab

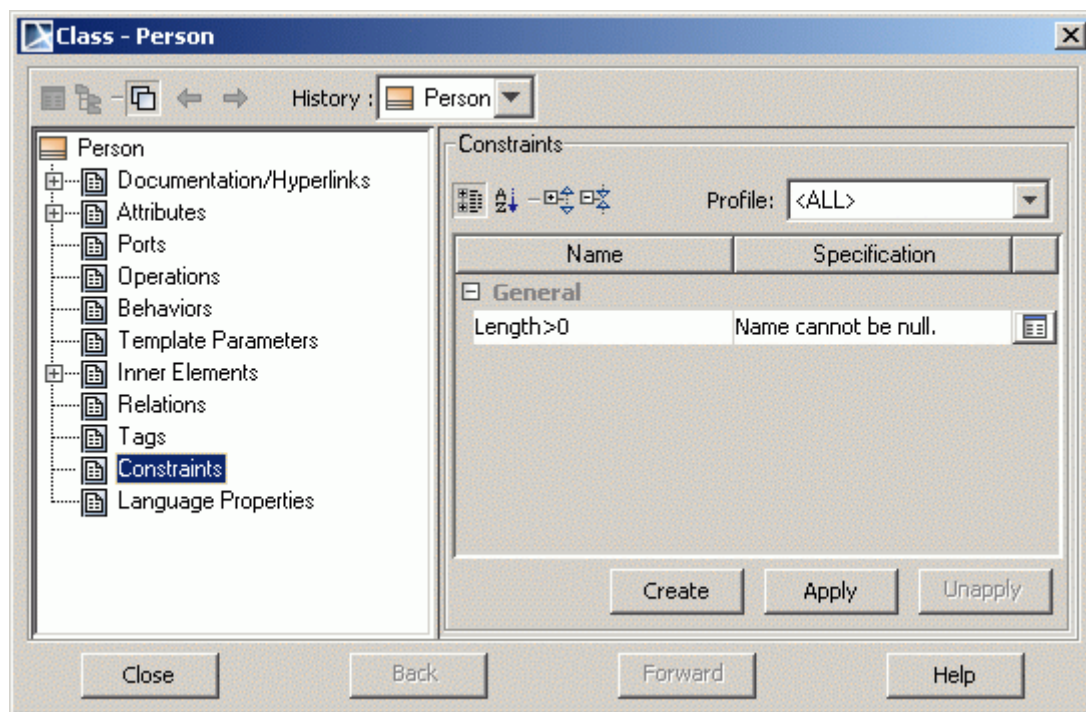



Figure 132 -- Specification dialog box. Constraints tab

Element name	Function
	The list of constraints assigned to the current model element.
Name	Enter the name of the constraint.
Specification	<p>A comment associated with the selected constraint. To edit the comment, double click the Specification line.</p> <p>Click the "...” button to open the Edit Specification dialog box. This allows you to edit expressions and select the Object Constraint Language (OCL) to check the expression syntax.</p>
	Click this button to open the Constraint Specification dialog box.

Element name	Function
Create	Creates a constraint.
Apply	The Select Elements dialog box opens. Select an existing constraint from the model and apply it to the element.
Unapply	Removes the selected constraint from the list.

Buttons available in the Specification dialog boxes



Button	Function
Close	Save changes and exit the dialog box.
Back	Return to the previous dialog box.
Forward	Proceed to the next dialog box.
Help	Display MagicDraw Help.

Default Property Values

MagicDraw version 15.0 and above allows for defining the initial (default) properties for elements.

The Default element properties can be defined for:

- the whole project.
- the specific diagram.

To set the default properties for the whole project

1. From the **Options** main menu, select the **Project** command. The **Project Options** dialog box opens.
2. Expand the **Default model properties** branch. Select the exact element and in the right pane side, change the property value.

After creating a new element it will have new property values. Values for previously created elements will not be changed.

To reset element properties to the default value, click the **Reset to Defaults** button. To reset property values for all elements select the **Default model properties** branch and click the **Reset to Defaults** button.

(Exception: interface attribute default visibility will always be #public, no matter what your settings.)

To set the default properties for the specific diagram

1. From the **Diagrams** main menu, select the **Customize** command. The **Customize Diagram** wizard opens.
2. Define the new or created diagram properties and in the **Specify toolbar buttons** step, click the **Add** button. In the appeared menu that opens, select the **New Button** command. The **Edit button** dialog box opens.
3. Open the **Element Properties** tab. Select the **Specify own values** radio button and change the default element property values.

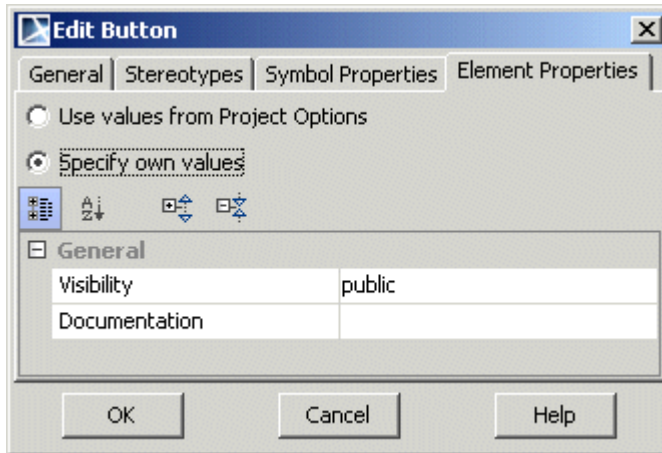


Figure 133 -- The **Edit Button** dialog box, **Element Properties** tab

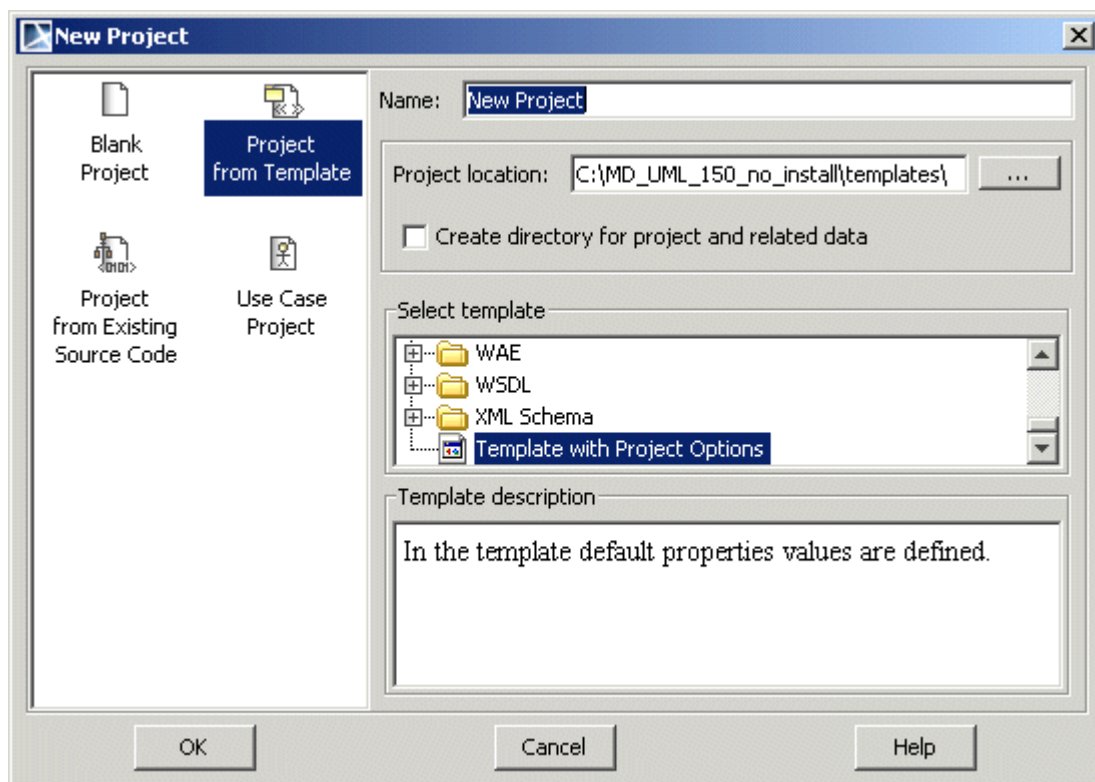
Create a new element from the customized diagram toolbar and the element will have the defined, default properties.

For more information about the **Customize Diagram** wizard, see *UML Profiling and DSL User-Guide.pdf*, which is located in the <MagicDraw installation directory>/Manual folder.

Sharing the default property values

If you want to share the default properties with other users for their new project, change the property values in the **Project Options** dialog box and then create a project template, which other users may use:

1. From the **File** main menu select the **Export > Template...** command and save the project as a template. A template will be created in the *<MagicDraw installation directory>/templates* directory.
2. To import the created template to a new project from the **File** main menu, select the **New Project** command. The **New Project** dialog box opens. Select the **Project from Template** icon and in the **Select template** tree, select your template. The project options are imported to the project together with the template.



Parent topic: "Working With Diagrams", on page 245.

Related topics:

“Project Options” on page 186.

Formatting Symbols

In MagicDraw, every symbol may have its own style: color, font, size, and so forth. There are several ways to define the symbol properties:

- From the symbol properties dialog box: from the selected symbol shortcut menu or from the **Edit** menu, select **Symbols** and then **Symbol(s) Properties**.
- From the **Project Options** dialog box: this dialog box enables you to change all available symbol properties, create your own style for the project, apply different symbol properties for different diagrams, define stereotype properties that may be bound to the symbol, and more.
- From the MagicDraw main toolbar: using this toolbar, you may change the color, font, and path style of a symbol.

For more information about symbol properties, see “Project Options” on page 186.

To set the symbol properties (fill color, text color, and others)

- From the symbol shortcut menu, select **Symbol(s) Properties**.
- From the **Options** menu, select **Project**. The **Project Options** dialog box opens. Select the desired options for the selected model elements.
- Select a symbol and from the **Edit** menu, select **Symbol**, and then select **Symbol(s) Properties**.

NOTE To apply changes made in the **Project Options** dialog box, click **Apply**.

Changing symbol properties dialog for multiple symbols

Symbol properties dialog may be invoked now when multiple symbols are selected in the diagram: select multiple symbols, invoke context menu and choose command **Symbol(s) Properties**.

Common selected symbols properties are displayed by default. If you wish to see all properties, turn on **Show All Properties** button that is located above properties table.

Note: Multiple symbol properties also may be changed using quick properties panel at the bottom of the Browser.

To show/hide model element constraints, stereotype and/or tagged values on the diagram pane

- Select/clear the **Show Constraints**, **Show Stereotype**, and/or **Show Tagged Values** check boxes from the symbol shortcut menu.
- 3. From the **Options** menu, select **Project**.
- 4. The **Project Options** dialog box opens. Select the **Show Constraints**, **Show Stereotype**, and/or **Show Tagged Values** check boxes for the selected model elements.

NOTE	To apply changes made in the Project Options dialog box, click the Apply button in the Styles tab.
-------------	---

Displaying icon or image on the symbol or instead the symbol

Definitions

Definition	Description
Symbol	The term "symbol" means a visual representation of some model elements in the diagram. Symbols are subdivided into shapes and paths (lines in the model, for displaying various relationships).
Symbol properties	Every symbol may have its own style: color, font, size, and so forth. Symbol properties may be defined for the concrete symbol, for all symbol of one element, or according to the diagram type. For more information about symbol properties definition, see "Formatting Symbols" on page 342, about style engine, see "Style Engine" on page 346.
Stereotype	"A stereotype defines how an existing metaclass may be extended. It enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass." [The OMG UML specification (UML 2.2: Superstructure)]. For more information about stereotype, see "Stereotype" on page 803. For more information about applying stereotype properties, see "Stereotype properties" on page 360.
Icon	Icon of stereotype. Icon is a small image displayed in the top-right corner of shape. To assign icon to stereotype in the Stereotype specification dialog box, define the Icon property. For more information on how to assign Icon for stereotype, see "To create a stereotype with an image" on page 803.
Text	Stereotype name, displayed on the symbol.
Image	Image which can be assigned to element and displayed as icon or instead of element shape. To assign image to element in the element specification dialog box, assign the Image property. For more information on how to assign icon for element, see "Image" on page 327.


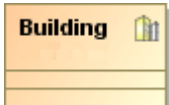



To change the icon visibility mode on the element shape:

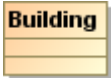
1. From the element shape shortcut menu, select the **Presentation Options > Show Stereotypes** command and then select the desired property mode.
2. You may change the stereotype/icon visibility mode in the symbol **Properties** dialog box > **Show Stereotypes** combo box.

6 WORKING WITH MODEL ELEMENTS

Formatting Symbols

Select one of the six property modes for **Show Stereotypes**. The property modes are described in the table below:

Show Stereotypes Property Mode	Displayed on the diagram pane	Icon of the stereotype and stereotype name are assigned to element	Image to element is assigned
Icon and Text		Icon of stereotype displayed. Name of stereotype displayed.	Image of element is displayed in the corner of shape.
Icon		Icon of stereotype displayed in the corner of symbol. Name of stereotype is not displayed.	Image of element is displayed in the corner of shape.
Text Only		Icon of stereotype is not displayed. Name of stereotype is displayed.	Image of element is not displayed.
Shape Image and Text*		Icon of stereotype is displayed instead of shape. Name of stereotype is displayed.	Image of element is displayed instead of shape.
Shape Image*		Icon of stereotype is displayed instead of shape. Name of stereotype is not displayed.	Image of element is displayed instead of the shape.

Show Stereotypes Property Mode	Displayed on the diagram pane	Icon of the stereotype and stereotype name are assigned to element	Image to element is assigned
Do Not Display		Icon of stereotype is not displayed. Name of stereotype is not displayed.	Image of element is not displayed.

* - To display the icon of a stereotype instead of the element shape all element compartments should be suppressed.

TIP! If element has assigned both - image and stereotype icon - then image of element will be displayed on the shape.

NOTE **Shape Image and Text** and **Shape Image** properties are not added to the Path element properties list.

Style Engine

The Style engine is a part of the MagicDraw UML system that defines diagrams, shapes, paths, and stereotype properties. There may be few property styles defined, but all symbols are created according to the style that is selected as default. There is a possibility to apply different presentation styles for diagram/shape/path/stereotype depending on the diagram type.

Symbol Property Styles Tree

Expands the tree hierarchy of all the styles defined within the project. You may use as many of these styles as you wish.

Shape and **Path** trees have the inner structure to help you find the model element, the representation of which must be changed. The right side of the dialog box contains possible choices and instruments to manage them.

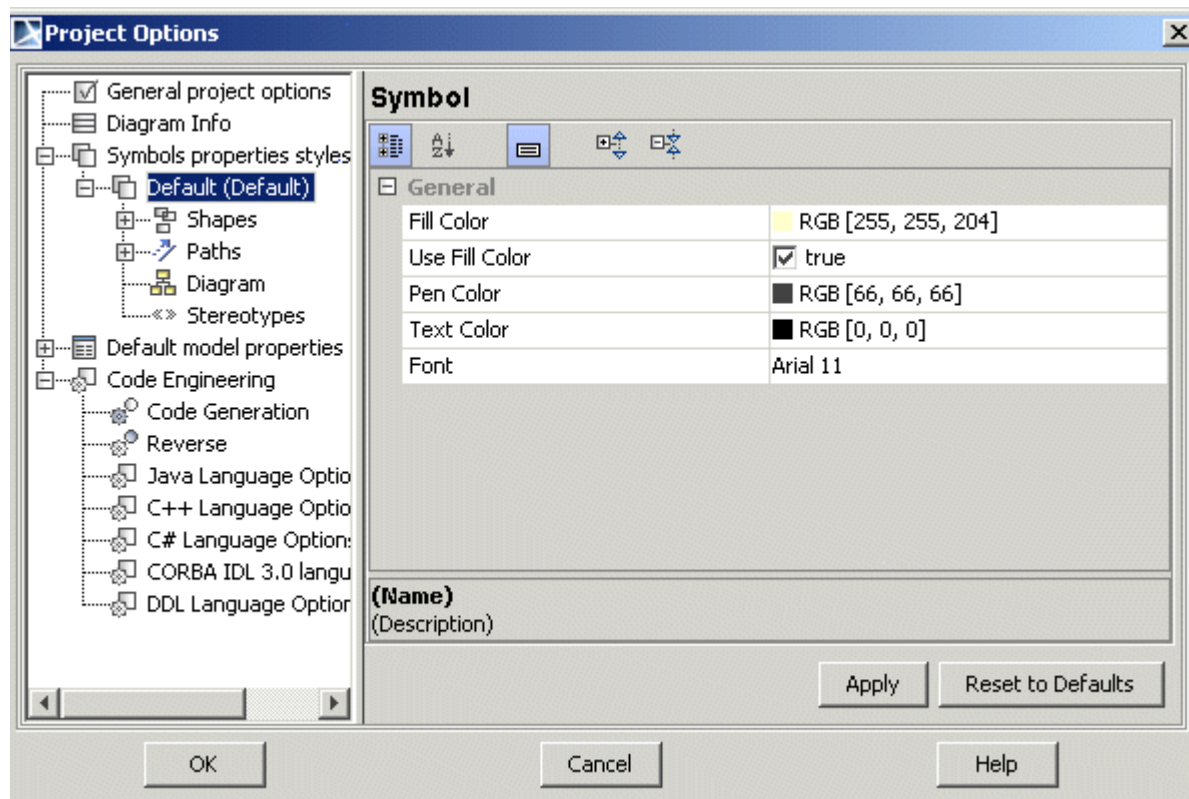


Figure 134 -- Project Options dialog box. Styles

The following buttons are available in the **Project Options** dialog box, the **Symbols properties styles** branch:

Button name	Function
Clone	Clone (duplicate) the selected style.
Rename	Change the name of all existing styles. Type a new name for a style in the Enter Style Name dialog box.
Delete	Remove the selected style.
Make Default	Make the selected style the default style for newly created projects.
Apply	Apply the selected style to the open project.

Button name	Function
Import	The “ Open dialog box” opens. Select the style you wish to import (*.stl).
Export	The “ Save dialog box” opens. Select the directory where you wish to export a style.
Reset to Defaults	Change all settings to the default configuration.

To create a new style by cloning the existing one

1. Select the default style in the **Styles** list box and click the **Clone** button.
2. Type a name for the new style in the **Enter Style Name** dialog box.
3. Change options of the new style.

To change the name of the selected style

1. Select a style you want to rename and click the **Rename** button.
2. Type a new name for the style in the **Enter Style Name** dialog box.

To remove the selected style

Click the **Delete** button in the **Project Options** dialog box.

To make a selected style your default style for newly created projects

Click the **Make Default** button in the **Project Options** dialog box.

To apply the selected style or changed option to a current project

- Click the **Apply** button in the **Project Options** dialog box, **Styles** pane.

NOTE

You can also apply the desired options to the selected diagram model elements. Click the **Apply** button in the specific elements pane.

To import an already created (and exported) project style


Click the **Import** button. The **Open** dialog box opens. Select the style you wish to import (*.stl).

To save the created style (export) for a later usage or for other users

Click the **Export** button. The **Save** dialog box opens. Select the directory where you wish to export a style.

In the following table you will find all possible options that can be set for the symbols:

Property	Function (when selected)
Fill Color	The fill color of the symbol. The Color dialog box opens.
Use Fill Color	Uses the selected fill color and the symbols color changes on the diagram.
Pen Color	The pen color that is used to draw elements. The Color dialog box opens.
Text Color	The color that is used for text coloring. The Color dialog box opens.
Font	The font that is used for the name and other displayed properties of a model element. The Font dialog box opens.
Autosize	Adjusts the size of a symbol to the contained information. Element borders are changed so that it uses minimum space.
Use Fixed Connection Points	The end of the path is connected to the fixed point of the shape.
Path Style	The drawing style of a path. Possible choices: Rectilinear , Oblique , or Bezier .
Attributes Color	The color of the attribute name. The Color dialog box opens.
Attributes Font	The font that is used for the name of an attribute. The Font dialog box opens.
Attributes Sort Mode	The mode for sorting attributes. Possible choices: No Sorting , By Name , By Stereotype , or By Visibility .
Constraint Text Mode	Displays constraint name or expression on a symbol.

Property	Function (when selected)
Show Qualified Name for Operation	<p>The Show Qualified Name for Operation property shows the operation name and the class of the operation on the activity shape (using style <i>ClassName::OperationName</i>). The default property value is true.</p> <div>  </div> <p>NOTE For projects that were created before MagicDraw version 15.0, the Show Qualified Name for Operation property is unchecked.</p> <p>For more information about displaying qualified name for operation, see "Call Operation Action" on page 829.</p>
Direction	<p>The direction of a signal.</p> <p>Possible choices: Right or Left.</p>
Enable Lolipop notation	<p>If the Enable Lolipop notation check box is selected, notation of the interface becomes as "lolipop".</p> <p>NOTE The Enable Lolipop notation property is included only for the interface symbol properties</p>
Header Position	<p>The package name position on the symbol.</p> <p>Possible choices: Top or In Tab.</p>
Header in Bold	Shows the name of the symbol in bold.
HTML Text	Activates the HTML editor for the text of a note and a text box.
Line Style	<p>A line style for a horizontal separator.</p> <p>Possible choices: Dashed or Solid.</p>
Operations Color	The color of the operation name. The Color dialog box opens.
Operations Font	The font that is used for the name of an operation. The Font dialog box opens.
Operations Sort Mode	<p>The mode for sorting operations.</p> <p>Possible choices: No Sorting, By Name, By Stereotype, or By Visibility.</p>
Orientation	Primarily the synchronization bar diagram button is set to the vertical or horizontal position.

Property	Function (when selected)
Show Attributes Constraints	Shows constraints of attributes.
Show Attributes Properties	Shows tagged values of attributes.
Show Attributes Stereotypes	Shows stereotypes of attributes.
Show Attributes Visibility	Shows attribute visibility signs (+, -, #, ~).
Show Base Classes	Shows a base class on the stereotype symbol.
Show Classifier	Shows a classifier name near the model element name.
Show Direction Arrow	Shows the Direction Arrow on the association. Default Direction Arrow direction is displayed according path creation direction. It helps to read diagram and explain diagram semantics. For more information about Direction Arrow, see "To show the direction arrow near the association name" on page 838.
Show Elements List	Shows model elements that are assigned to a model, package, or subsystem as a list.
Show Entire Activation	Shows the entire activation bar on an active classifier role in a sequence diagram.
Show Full Classifier Type	Shows all attributes that are defined within a class or assigned classifier.
Show Initial Attribute Value	Shows the initial attribute value on a class or artifact.
Show Message Numbers	Shows the message numbers on a diagram.
Show More Sign for Attributes	Shows an additional information sign "..." in the class, artifact attributes list, when omissions are made by editing the class compartment.
Show More Sign for Operations	Shows an additional information sign "..." in the class, artifact or enumeration operations list, when omissions are made by editing the class, artifact, or enumeration compartment.
Show Multiplicity	Shows the multiplicity value.
Show Name	Shows the name of a relationship, role and message/stimulus.
Show Operations Constraints	Shows constraints of operations.
Show Operations Signature	Shows all of the operation arguments and the return type.
Show Operations Stereotypes	Shows stereotypes of operations.
Show Operation Parameters Direction Kind	Shows the direction kind for operation parameters
Show Operations Properties	Shows tagged values and concurrency of an operation.

Property	Function (when selected)
Show Operations Visibility	Shows operation visibility signs (+, -, ~ #).
Suppress Signal Receptions	Hides attributes list from the shape.
Signal Receptions Color	The color of the signal reception name. The Color dialog box opens.
Signal Receptions Font	The font that is used for the name of a signal reception. The Font dialog box opens.
Show Signal Receptions Signature	Shows the parameter and related information.
Show More Sign for Signal Receptions	Shows additional information sign “...” in the signal reception list, when omissions are made by editing the signal reception compartment.
Signal Receptions Sort Mode	The mode for sorting signal receptions. Possible choices: No Sorting , By Signal Name , By Stereotype , or By Visibility .
Show Signal Receptions Visibility	Shows signal reception visibility signs (+, -, ~ #).
Show Signal Receptions Stereotypes	Shows stereotypes of signal receptions.
Show Signal Receptions Properties	Shows tagged values and concurrency of properties.
Show Signal Receptions Constraints	Shows constraints of signal reception.
Show Signal Receptions parameter Direction Kind	Shows the direction kind for signal reception parameters.
Show Owner	Changes the display position of qualified name on the element shape.
Show Predecessors	Shows predecessors on the message.
Show Visibility	Shows role visibility signs (+, -, #).
Stereotype Color	The color that will be used to draw stereotypes. The Color dialog box opens.
Stereotype Font	The font that will be used to draw stereotypes. The Font dialog box opens.
Show Stereotypes	If you want to change a stereotype and its icon visibility on the element shape, use the Show Stereotypes property. For more information about stereotype display mode, see “Changing the stereotype display mode” on page 809.
Show Constraints	Shows constraints on symbols.

Property	Function (when selected)
Show Tagged Values	Shows tagged values on symbols.
Suppress Actions	Hides actions associated with the state.
Suppress Attributes	Hides the attribute list.
Suppress Extension Points	Hides use-case extensions on a use case.
Extension Points Color	The color of the extension point name. The Color dialog box opens.
Extension Points Font	The font that is used for the name of an extension point. The Font dialog box opens.
Suppress Enumeration Literals	Hides enumeration literals on a enumeration.
Enumeration Literals Color	The color of the enumeration literal name. The Color dialog box opens.
Enumeration Literals Font	The font that is used for the name of an enumeration literal. The Font dialog box opens.
Suppress Operations	Hides operations compartment section.
Suppress Realization Elements	Hides realization elements of a subsystem.
Suppress Specification Elements	Hides specification elements of a subsystem.
Text Position	Changes the text position of a separator. Possible choices: Center , Left , or Right .
Wrap Words	Wrap words to a new line when text exceeds the textbox width.
Background Color	The color of the diagram background. Click the “...” button to open the Color dialog box, select the background color.
Snap Paths to Grid	Snap paths to grid.
Snap Shapes to Grid	Snap shapes to grid.
Grid Size	Grid size settings from 2 to 30.
Show Grid	Shows a grid on the diagram.
Show Diagram Info	Shows diagram information table on the diagram pane.

Working with Properties Styles

All symbols in MagicDraw are created according to active properties styles. There may be more than one property style in the same project, and the whole style may be applied for the project.

Every style has its own presentation of Diagram, Shape, Path, and Stereotype that you can modify using the **Project Options** dialog box in the **Symbols Properties Styles** branch. You can set your own options for every model element to the current style.

Path, Shape, and Stereotype branches have the inner structure that helps you find the model element, the representation of which must be changed. The section on the right side of the dialog box contains possible choices and instruments to manage them.

The following properties are defined for the formatting symbols:

- **Shapes.** Set general options for the shapes in the right pane of the **Project Options** dialog box. You can set options for all shapes that appear on the Diagram pane.
- **Paths.** Set general options for the paths in the right pane of the **Project Options** dialog box. You can set options for all paths that appear on the Diagram pane.
- **Diagram.** Set general options about a diagram.
- **Stereotypes.** Set general options for the stereotypes in the right pane of the **Project Options** dialog box. You can set options for all stereotypes that may be applied to elements on the Diagram pane.

Changing properties for multiple element

To change properties for multiple symbols, using Ctrl or Shift key select few elements in the **Project Options** dialog box, **Symbol properties styles** branch.

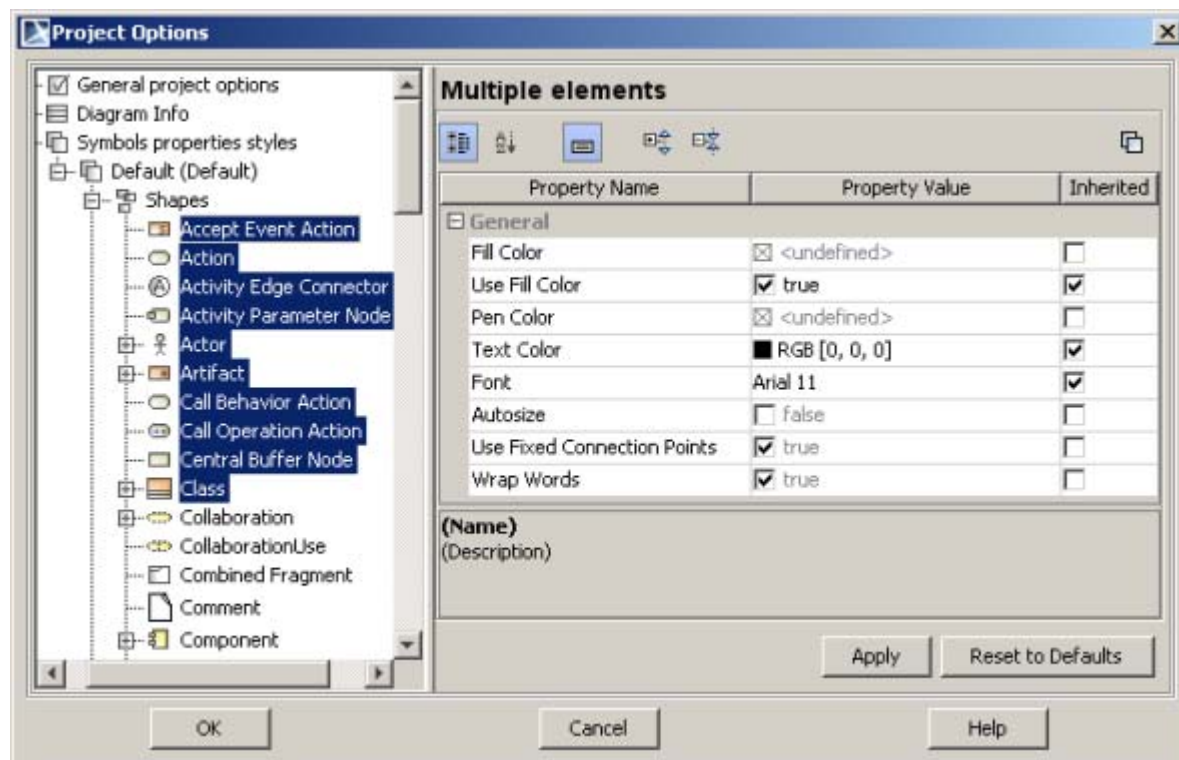


Figure 135 -- The Project Options dialog box, multiple element symbols style is selected

Properties extension by diagram

Diagram, shape, path, and stereotype properties can be extended by the particular diagram type. This means that presentation style options will be applied only for the specified element symbol in the specific diagram.

To extend the element properties by diagram

1. In the **Project Options** dialog box, the **Symbols Properties Styles** tree, expand a branch, select the specific element (shape, path, diagram, or stereotype) and right click the mouse button. The list of diagrams in which the element symbol may be created, opens.

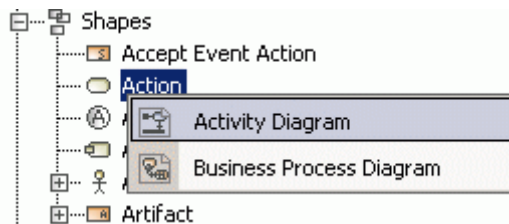


Figure 2 -- Element extension by diagram

2. Select the diagram type. The Diagram is added as an additional branch to the section.

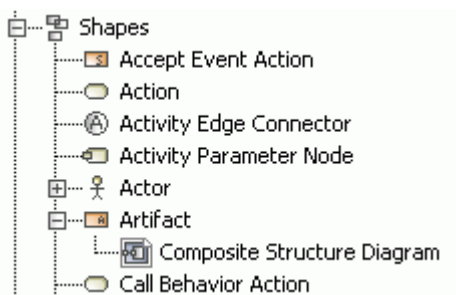


Figure 3 -- Extended diagram addition

3. Set the style properties for the element in the right pane of the **Project Options** dialog box. The properties will be applied only in the specified type of diagram.

- The element can be extended by diagram in the **Project Options** dialog box, specific elements pane, by clicking the **Extend by Diagram** button. The **Extend by Diagram** dialog box opens. Click the **Add Diagram** button and select a diagram from the list.

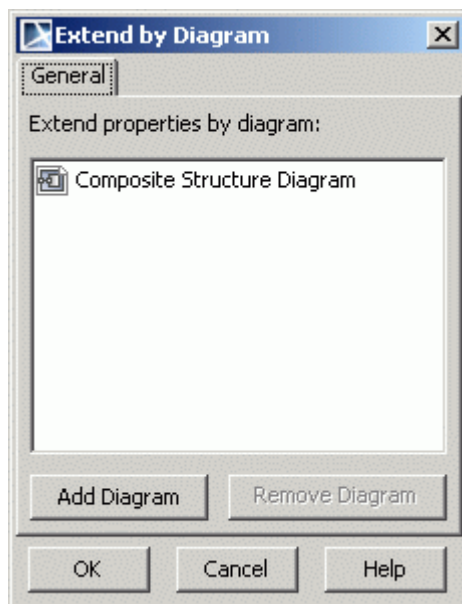


Figure 4 -- Extend by Diagram dialog box

To remove the extended diagram from the tree

- Select the extended diagram and right click on the mouse, then select **Remove**.

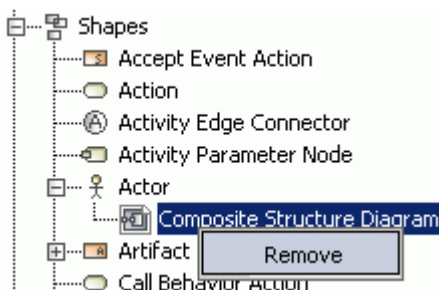


Figure 5 -- Remove extended diagram

- In the **Project Options** dialog box, the extended diagram style properties pane, click the **Remove** button.

Properties Inheritance

All element properties have the “inherited” check box. This check box indicates the property is derived from the base element properties or is it specific.

The **Inherited** column check box value in the elements properties pane specifies if the current property is synchronized with its parent property. When the **Inherited** value is “true”, the element property is changed after changing the parent property.

If the property has no correspondent property in the upper (parent) level, the **Inherited** column check box will be cleared and disabled.

If the property is modified for the specific element and the value differs from the upper level current property value, the **Inherited** column check box is cleared automatically.

General Style Properties

You can define the common properties for the whole style. The style properties are displayed when the properties style is selected in the **Project Options** dialog box styles tree.

Shape, Path and Diagram Properties

All shapes, paths and diagrams that can be created in the project, are listed in the **Project Options** dialog box. If the **Shape**, **Path**, or **Diagram** branches are selected in the tree, the general properties can be set in the right pane of this dialog box.

When expanding any of these branches, the style for a concrete element (diagram) can be created.

To apply a new style to a previously created element symbol

1. In the **Project Options** dialog box, change the element style properties and click the **Apply** button. The **Select Diagrams** dialog box opens. The list of diagrams created in the project is displayed.

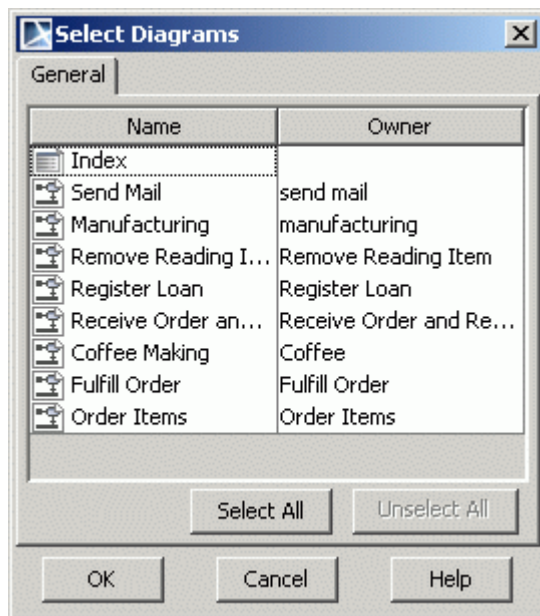


Figure 6 -- Select Diagrams dialog box

2. Select the diagrams to which the element properties will be applied and click **OK**. The **Select Properties to Apply** dialog box opens.

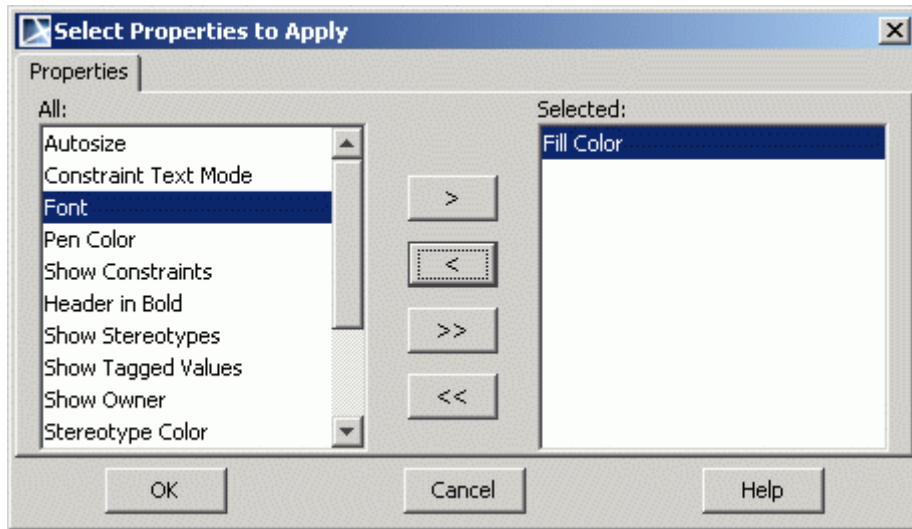


Figure 7 -- Select Properties to Apply dialog box

3. Select the properties to be applied to the element symbol by moving them from the **All** list to the **Selected** list. Click **OK**.

NOTE!

If a new style was set, it will be applied for all newly created elements after drawing them on the diagram pane. You can always set the default symbols style to the element by clicking the **Apply Default Symbol Style** button in the main toolbar.

Stereotype properties

The Stereotype properties can be applied only if the stereotype properties style is created in the **Project Options** dialog box.

The Stereotype properties are derived from their base class. The Stereotype base class is defined in the label of the right pane of the **Project Options** dialog box.

The same element can have several stereotypes assigned. In this case, the style of the first stereotype will be applied to the element symbol. If the stereotype is removed from the element, the next (first) stereotype properties are applied. If the last stereotype is removed from the element, the base class (shape or path) properties are applied to the element symbol.

Stereotypes may be extended by diagram.

All stereotypes that have defined symbol properties are included in the **Stereotypes** branch. By default only *boundary*, *control*, and *entity* stereotypes are added to the tree when expanding the **Stereotypes** branch. The default style is created for these stereotypes.

To add a stereotype to the branch

1. In the **Project Options** dialog box, the **Symbols Properties Styles** tree, right-click the **Stereotypes** branch. The list of stereotypes opens.

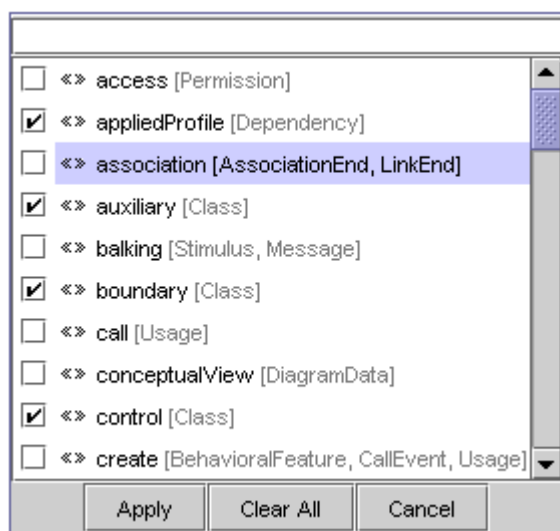


Figure 8 -- List of stereotypes

2. Select the check box near the stereotype and click the **Apply** button. The stereotype will be included into the **Stereotypes** branch. Set the stereotype style properties in the right pane of the **Project Options** dialog box.

To remove a stereotype from the branch

1. In the **Project Options** dialog box, the **Symbols Properties Styles** tree, select the **Stereotypes** branch. The list of stereotypes opens.
2. Clear the check box near the stereotype and click **Apply**. The stereotype is removed from the branch.

To change stereotype properties

1. Expand the **Stereotypes** branch and select a stereotype.
2. Set properties in the right pane of the **Project Options** dialog box.

To apply the stereotype properties to a previously created symbol with assigned stereotype

1. When the stereotype style properties in the right pane of the **Project Options** dialog box are changed, click the **Apply** button. The **Select Diagrams** dialog box opens.
2. Select the diagrams to which the stereotype properties will be applied and click **OK**.
3. In the **Select Properties to Apply** dialog box, select which properties will be applied to the stereotype. Click **OK**.

NOTE!

You can apply stereotype properties to a symbol after changing the style properties and in the **Project Options** dialog box, click **OK**. The style will be applied when selecting symbol on the diagram pane and clicking the **Apply Default Symbol Style** button on the main toolbar .

To apply the stereotype properties when assigning a stereotype to an element

NOTE!

In the **Environment Options** dialog box, **Diagrams** tab, the **Apply Stereotype Style for All Symbols** check box should be selected and in the **Project Options** dialog box, the style properties should be changed for stereotype.

1. In the created diagram, draw an element.
2. From the element shortcut menu, select **Stereotype**. The list of available stereotypes opens.
3. Select the check box near the stereotype you want to assign to the element. Click **Apply**. The stereotype properties are applied automatically when assigning the stereotype to the element.

Defining Hyperlinks Between Elements

You can set text for notes, text boxes, or separators as HTML text. You can also hyperlink to any model element, diagram, external file, or requirement.

Adding a hyperlink to the model element

There are three ways to add a hyperlink to the model element: from diagram **Smart Manipulator**, from **Specification** and from the Browser tree.

To add a hyperlink from the diagram

1. Select the element and click on the Smart Manipulator for Hyperlink.

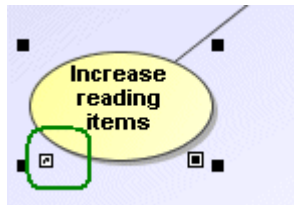


Figure 9 -- Smart manipulator for hyperlinks.

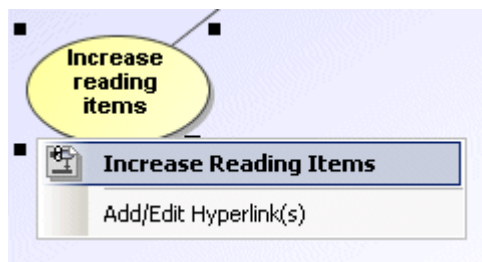


Figure 10 -- Hyperlinks menu.

2. The hyperlink menu opens, listing:
 - Previously created hyperlinks with icons corresponding to element type, diagram type, external file, or requirement.
 - Menu item for adding and editing existing hyperlinks - **Add/Edit Hyperlink(s)**
 - If there are no hyperlinks yet defined, only the menu item to add a hyperlink will be in the hyperlink menu.

3. Click **Add/Edit Hyperlink(s)** item. The hyperlinks creation editing dialog opens.

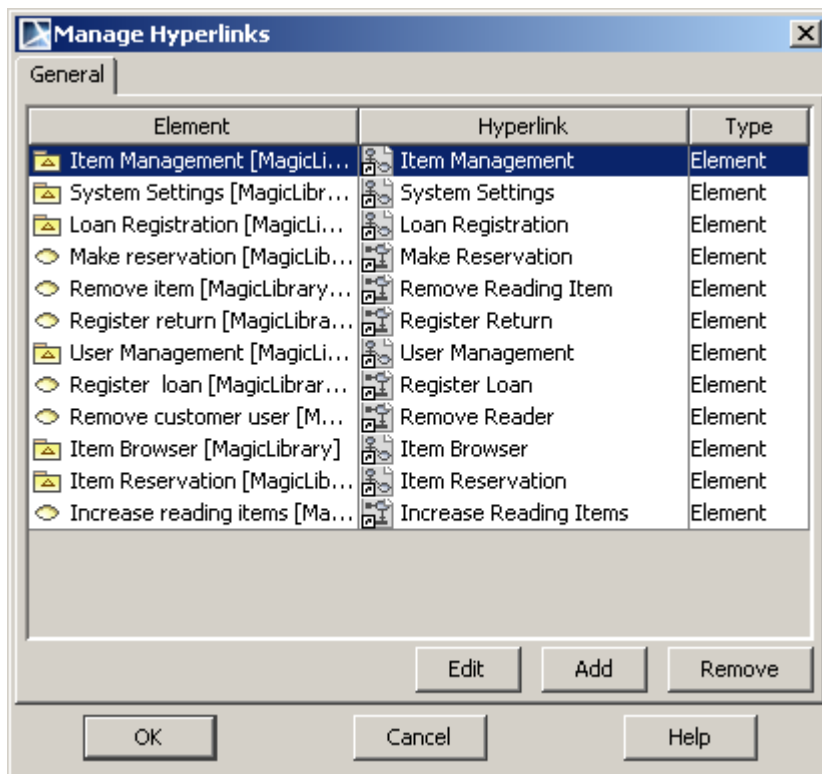


Figure 136 -- Manage Hyperlinks dialog

4. Click **Add** and define the hyperlink to any model element, file, or Web page in the **Edit Hyperlink** dialog box. If you want this hyperlink to be active, select the **Active** check box.
5. Click **OK**.

To add a hyperlink from Specification:

1. Open the model element **Specification** dialog box and select the **Documentation/ Hyperlinks** tab.
2. Define the hyperlink to any model element, file, or Web page in the **Edit Hyperlink** dialog box. If you want this hyperlink to be active, select the **Active** check box.
3. Click **OK**.

6 WORKING WITH MODEL ELEMENTS

Defining Hyperlinks Between Elements

To add a hyperlink from the Browser tree

Now hyperlinks can be created and edited straight from the element shortcut menu:

- Invoke element shortcut menu from the Browser.
- Choose **Go To** and **Hyperlinks** (see Figure 137 on page 365).

For more detailed description on managing hyperlinks, see “To add a hyperlink from the diagram” on page 363.

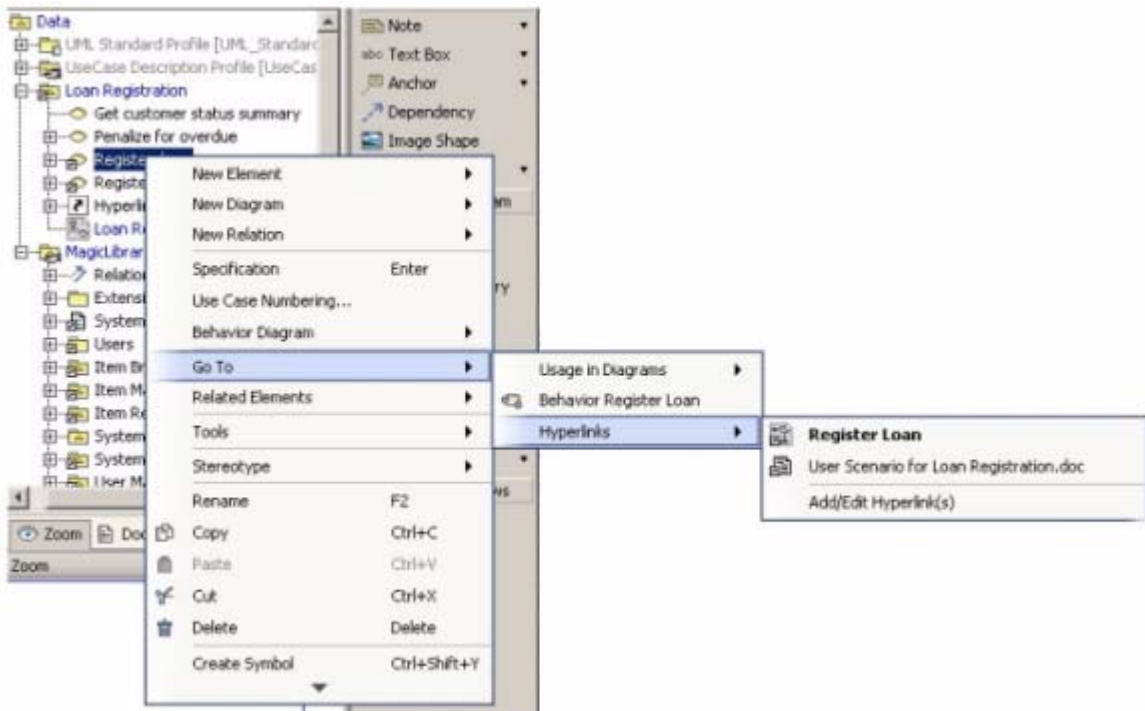



Figure 137 -- Hyperlinks creation from the Browser tree

To add a hyperlink to a note, text box, or separator text

1. Select the text where you want to add a hyperlink and click **Insert Hyperlink**  .

2. The **Edit Hyperlink** dialog box opens. Select the hyperlink you want to insert, either to a Web page, another model element, or a file:

- To link to an existing model element, click the **Element/Symbol** tab. Click the **Select Element/Symbol “...”** button and select the model element or symbol you want to link to in the **Select Model Element/Symbol** dialog box.
- To link to an existing Web page, click the **Web Page** tab, and in the **Type the Web page name** text box, type the URL of the Web page you want to link to. You can click the “...” button and browse the Web for the desired Web page.

NOTE

Set the path of the HTML viewer in the **Environment Options** dialog box (for a description, see “Setting Environment Options” on page 129.)

- To link to an existing file, click the **File** tab and enter the path to the file you want to link to. Or, click the **Type the file name “...”** button and, in the **Open** dialog box, select the file you want to link to.

NOTES:

- The selected file opens in the HTML browser.
- Set the path of the HTML viewer in the **Environment Options** dialog box.
- You can only link to an existing file. New files are not created for you.

Using the HTML editor toolbar, you can change the font, color, size, and the alignment of the selected text.

Edit Hyperlink dialog box

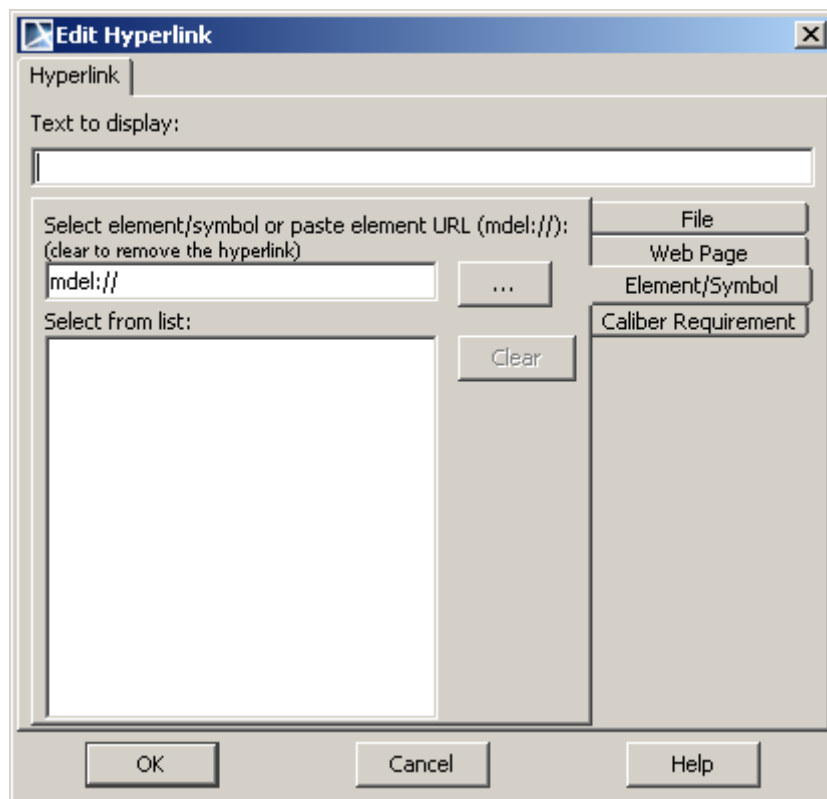


Figure 138 -- The Edit Hyperlink dialog box

6 WORKING WITH MODEL ELEMENTS

Defining Hyperlinks Between Elements

Tab name	Box	Function
Element/ Symbol Creates a hyperlink that goes to the selected model element.	Text to display	A text that will be displayed as a hyperlink.
	Select Element / Symbol or paste element URL“...”	The Select Model Element/Symbol dialog box opens. Select the model element you want to link to. You can also paste URL to element. For more information about element URL, see “Copying/ Opening Element URLs” on page 385.
	Select from list	A list of all items that have previously been selected as links.
	Clear	Remove all items from the Select from list .
	Active	If selected, activates the hyperlink on the diagram. Only one hyperlink can be active. Target referenced by the active hyperlink is accessed after double clicking an element with a hyperlink. By default the last added hyperlink is the active one.
Web Page Creates a hyperlink that goes to the specified Web page.	Text to display	The text that will be displayed as a hyperlink.
	Type the Web Page name “...”	Type the web page URL. Click the “...” button. The Web browser window opens. Browse the Web and find the web page you want to link to. NOTE Make sure that the path of the HTML viewer is set in the Environment Options dialog box.
	Select from list	A list of all items that have previously been selected as links.
	Clear	Remove all items from the Select from list .

6 WORKING WITH MODEL ELEMENTS

Owner of the model element

Tab name	Box	Function
File Creates a hyperlink that goes to a specified file.	Text to display	A text that will be displayed as a hyperlink.
	Type the file name “...”	Type the path to the file you want to be opened or click the “...” button. The Open dialog box opens. Select the file you want to link to.
	Select from list	A list of items that have previously been selected as links.
	Clear	Remove all items from the Select from list .

Owner of the model element

Model elements and diagrams belong to a package, model (system boundary), subsystem or other appropriated model element, which is called *owner*.

The name of the owner is displayed in the model element name compartment in parentheses.

To add a model element to a package, model (system boundary), or subsystem

- Drag a model element to the desired package on the Diagram pane or in the Browser tree.
- Open the **Inner Elements** tab, located in the **Package**, **Model**, or **Subsystem Specification** dialog box. Click **Add** and select a model element or diagram you want to add to a package. Define a model element or diagram in the open **Specification** dialog box and click **OK**.
- From the selected owner shortcut menu in the Browser tree, select **New Element**. From the list, select the desired model element and type its name in the Browser.

To display/hide the package/system boundary/subsystem name (the owner of an actor) on a model element

- From the symbol shortcut menu, select **Symbol(s) Properties**. The **Properties** dialog box opens. Select/clear the **Show Owner** check box.

6 WORKING WITH MODEL ELEMENTS

Owner of the model element

- From the **Options** menu, select **Project**. The **Project Options** dialog box opens. Select the desired model element and select/clear the **Show Owner** check box. If you want to apply changes for previously created model elements, click **Apply**.

TIP!

For a class, actor, or interface, you may display/hide the name of the owner from the symbol shortcut menu: select **Presentation Options**, and then select/clear the **Show Owner** check box.

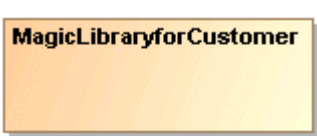
Owner display mode

MagicDraw version 15.0 and later has improved the owner display functionality. Now you can change the display position of the qualified name on the element shape.

To change the qualified name position:




- From the element shape shortcut menu, select the **Presentation Options > Show Owner** command and then select the desired property mode.
- You can change the qualified name position in the symbol **Properties** dialog box, **Show Owner** drop-down list.

Select one of the four property modes for **Show Owner**. The property modes are described in the table below.

Show Owner Property Mode	Shape	Description	Notation
Do Not Display		Only element name is displayed on the element shape. This is the default value.	-

6 WORKING WITH MODEL ELEMENTS

Owner of the model element

Show Owner Property Mode	Shape	Description	Notation
Below Element Name		Owner is displayed below the element name.	This is MagicDraw style notation. The owner name is constructed from the names of the containing namespaces starting at the root of the hierarchy and ending with the owner of the NamedElement itself. Containing namespaces are separated by dot and owner is displayed in brackets.
In Same Line With Name		Element owner is displayed in the same line as the element name.	This is a notation from UML specification. The qualified name is constructed from the names of the containing namespaces, starting at the root of the hierarchy and ending with the name of the NamedElement itself. Containing namespaces are separated by double colons. The double colon is shown to separate containing namespaces and element name.
Above Element Name		Owner is displayed above the element name.	Notation is the same as In Same Line With Name option notation.

Qualified name starting from model library

MagicDraw version 15.0 and above includes an option to show the owner hierarchy starting from the model library as the root.

Model Library is a package with modelLibrary stereotype.

This option is called **Qualified name display style**. To change its value:

1. From the **Options** menu select the **Project** command. The **Project Option** dialog box opens.
2. Select the **General Project Options** branch. In the right side pane, you can modify the option property.

The **Qualified name display style** property is added to the **Project Options** dialog box, **General Project Options** branch.

If the **Model Library Relative** property value is selected (default value for a new project), then the full qualified name hierarchy is displayed on the shape, starting from the model library as a root. The model library itself is not displayed. The **Qualified name display style** property allows for having the relative path for library items used in the project.

Relations Changes Ownership when Client or Supplier is Moved to Other Owner

Some issues related to relationships have been addressed to improve usability.

Now relationships will not get lost in Containment tree while changing the element ownership. Relationships will also be moved together with the client or supplier (or both) so that all the related elements can be grouped together in one place. This will also prevent unexpected dependencies on model partitioning.

For example, if you move two Classes which are connected to the Association relationship to another Package in the Containment tree, a question dialog will open, asking if you want to move the relationship as well (Figure 139 on page 372).

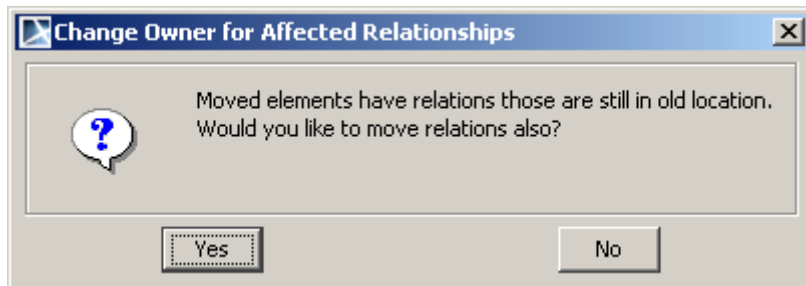


Figure 139 -- The Change Owner for Affected Relationships message

Converting An Element

Element conversion allows converting one element type to another. Sometimes, during the modeling process, there is a demand to change a class to a component or another type of classifier. The element conversion copies all compatible properties to a converted element (for example Ports of a class will become Ports of a component). If some properties are not compatible, they will be lost.

Element conversion functionality allows UML element conversion from one meta-class to another.

To convert an element

From the element shortcut menu, select **Refactor** and then **Convert To** and then select the element from the open list, to which you want to convert.

When an element is being converted, the converter finds all usages of this element and recreates these references to a new element after conversion. For example, if an instance specification has an element assigned as a classifier, it remains after conversion.

The usages that are not valid after conversion, will be removed. For example, an Interface has an Interface Realization relationship. An interface is converted to a Class. The Interface Realization will be removed from model.

If the converted element and new element have the same symbol properties, then they are reassigned for the new element. For example, if a class is converted to an interface, and the class had the property **Suppress Operations** - *true*, then the Interface property for **Suppress Operations** will be *true* also.

Replacing An Element

You can replace one model element with another of the same metatype type element. Model element replacement is useful when during the modeling process you notice that one model element needs to be replaced with another. All relations and references to former element are updated to point to the newly selected model element.

After the replacement source will be replaced with replacement target:

- All references to replacement source will be replaced by references to replacement target.

6 WORKING WITH MODEL ELEMENTS

Replacing An Element

- The replacement target will be displayed in all diagrams instead of replacement source.
- The replacement target after replacement will have all paths of replacement target and replacement source.
- Replacement source will be deleted.

To replace one element with another:

1. In the element shortcut menu, select **Refactor** and then **Replace With**. The **Select Element** dialog box opens.
2. Select the element with which you want to replace.

See an example in the Figure 140 on page 374 and Figure 141 on page 375. In the Figure 140 on page 374 you can see two classes, which are similar to each other - *Customer* and *User*. Using *Replacement functionality* you can replace *User* class with *Customer* class quickly without redrawing relationships (see Figure 141 on page 375).

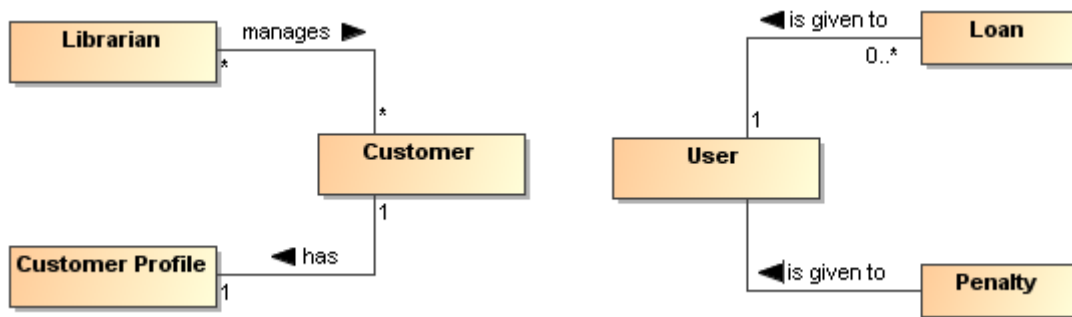


Figure 140 -- Model before the element replacement

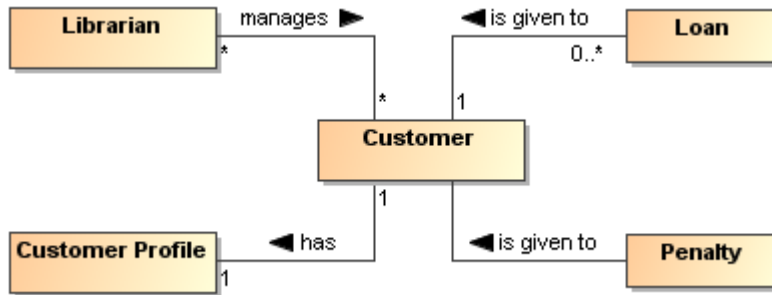


Figure 141 -- Model after the User class has been replaced with the Customer class

HTML Editor

MagicDraw has been enhanced with a new HTML editor to edit HTML text. The new editor improves text editing capability and usability and allows you to preserve the text format when copying formatted text.

To set the note/text box/separator text as HTML

- Select the element and click on Smart Manipulator *Switch To HTML Text* (Figure 142 on page 375).

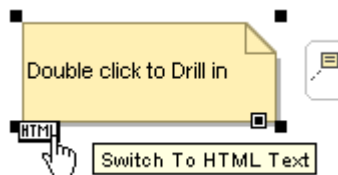


Figure 142 -- The Switch To HTML Text Smart Manipulator

- From the note/text box shortcut menu, select **HTML Text**.
- Draw the note or text box, using the **Note (HTML text)** or **Text Box (HTML text)** toolbar buttons.

When selecting the text, the HTML editor toolbar opens (Figure 143 on page 376).



Figure 143 -- HTML toolbar

To set text as HTML in dialog boxes

You can write HTML text in various dialog boxes. To turn on the HTML mode, click the HTML check box. See samples of HTML check box and HTML toolbar in dialog boxes:

- Element specification dialog box, documentation/hyperlinks tab (Figure 144 on page 377).
- In the Browser, Documentation tab (Figure 145 on page 378).
- Element specification dialog box, tags branch, when created tagged value (Figure 146 on page 379).
- Then specifying To Do property from the element specification dialog box (Figure 147 on page 380).
- And in other locations.

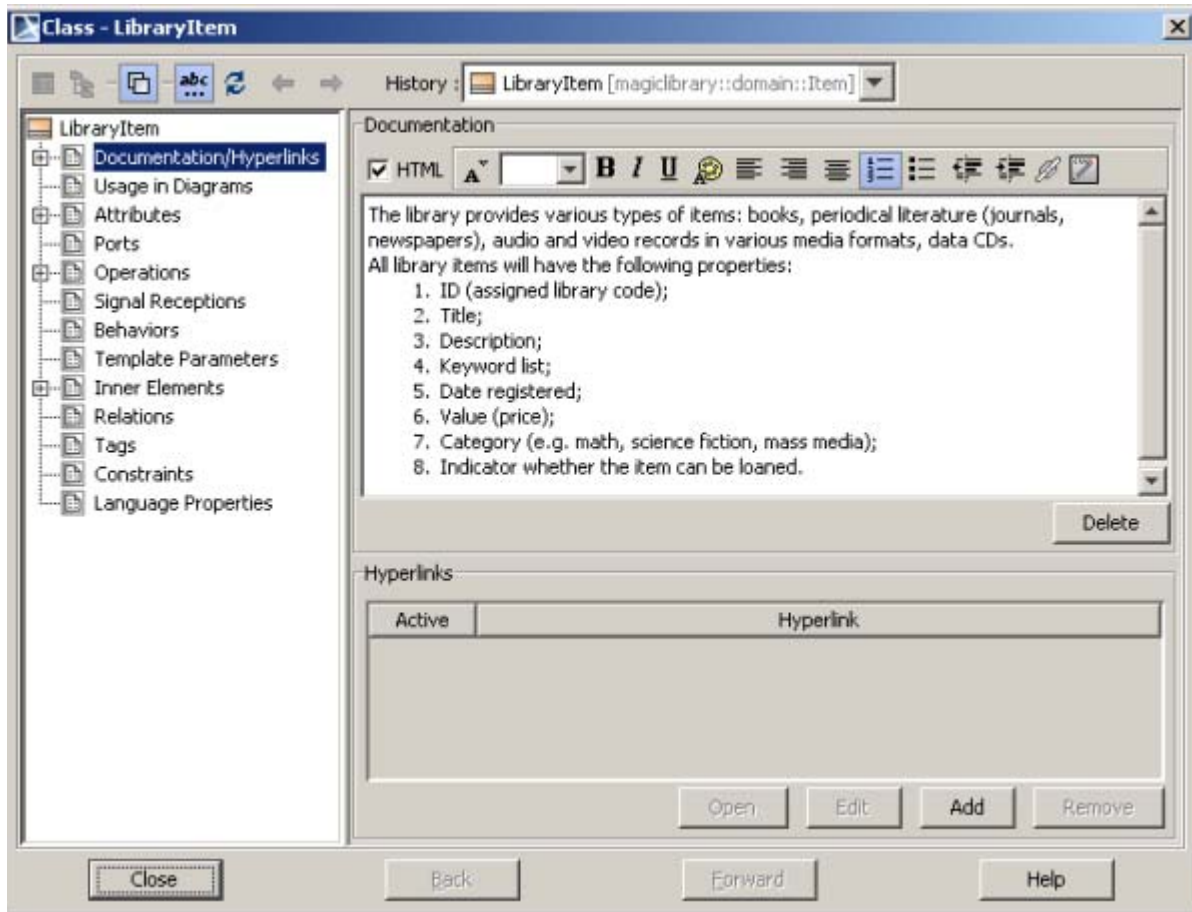


Figure 144 -- HTML toolbar in the Specification dialog box, Documentation/Hyperlinks branch

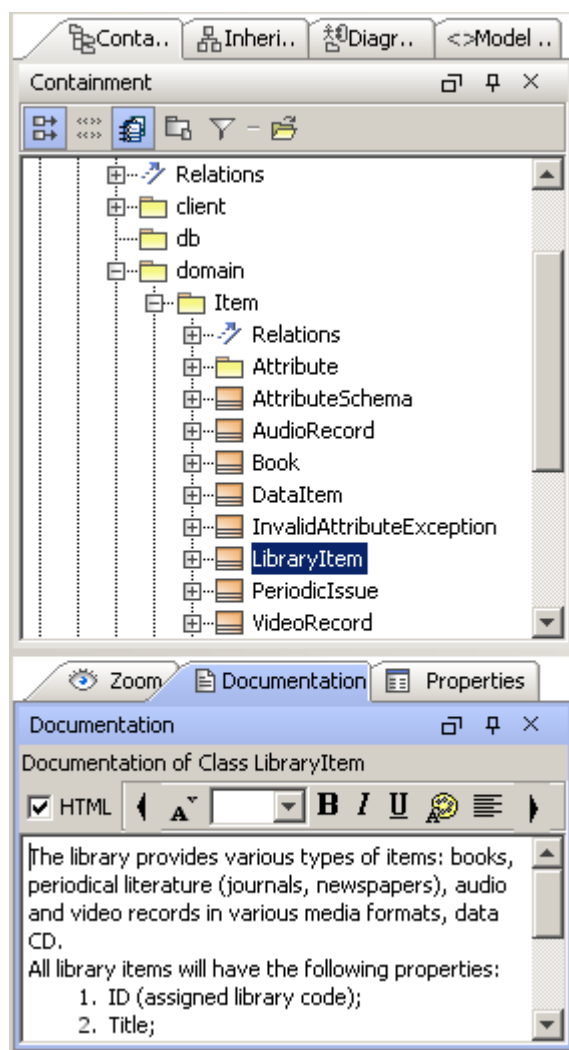


Figure 145 -- The HTML toolbar in the Documentation tab, in the Browser

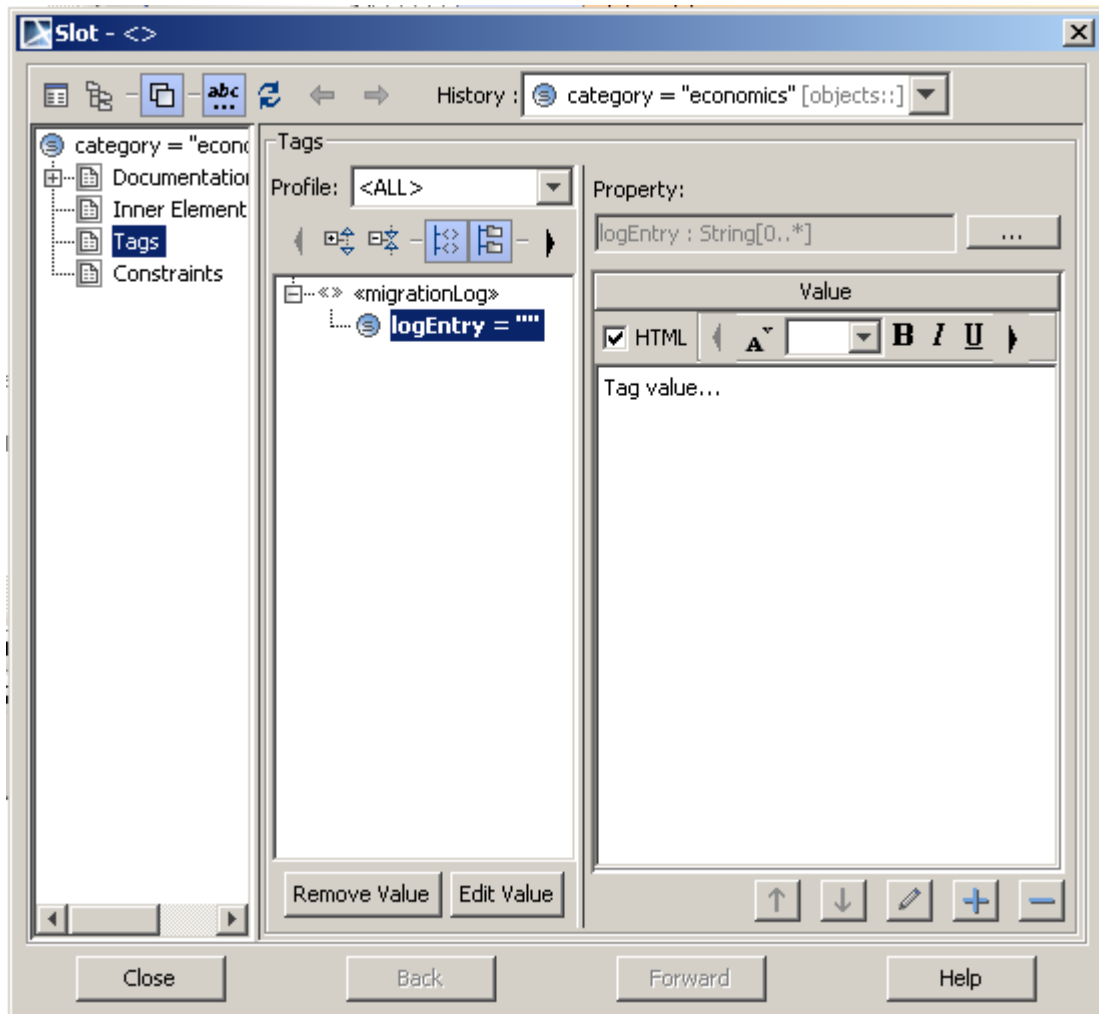


Figure 146 -- The HTML toolbar when editing Tag value

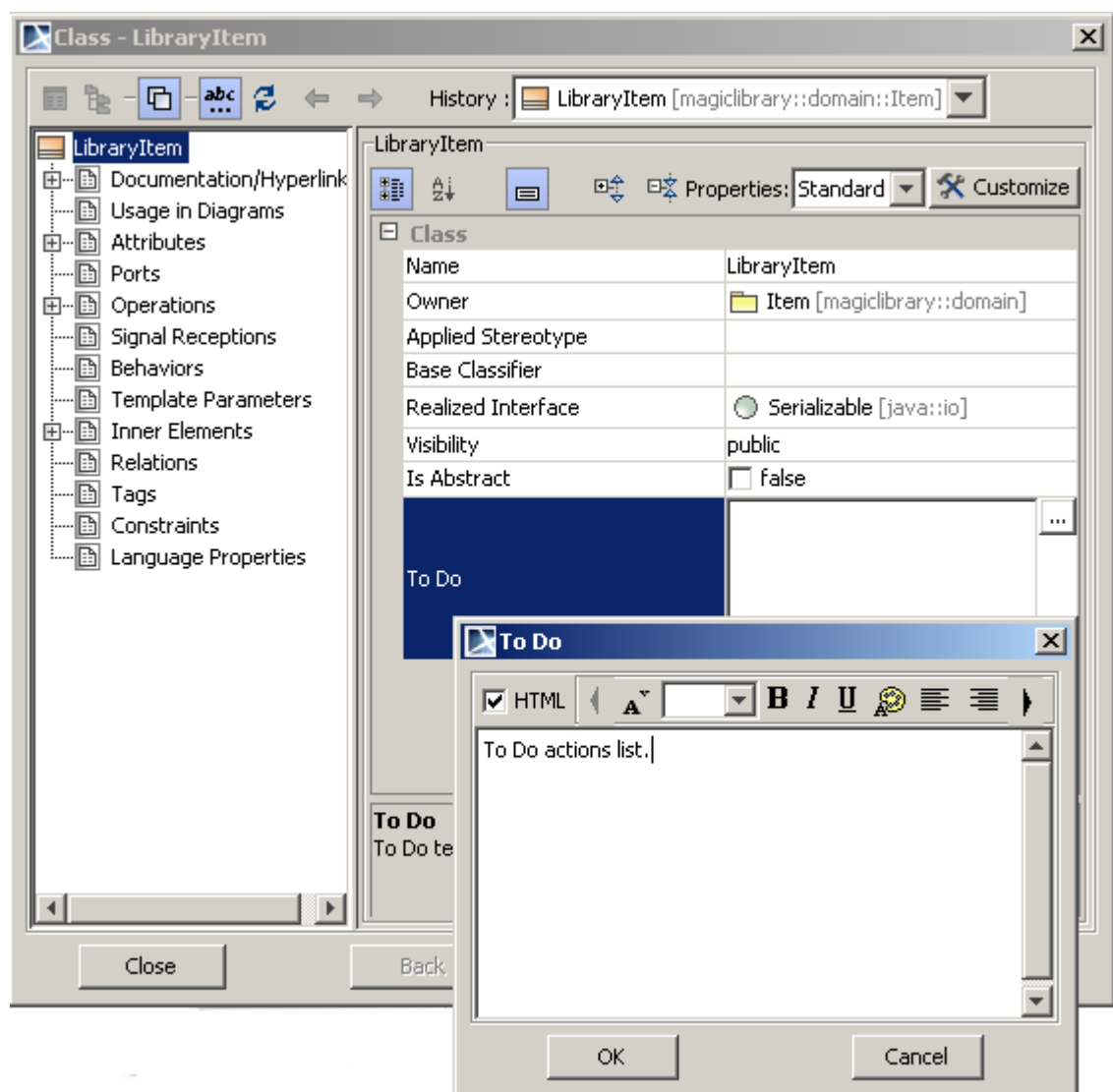


Figure 147 -- The HTML toolbar in the To Do dialog box, invoked from element specification dialog box



Editing an element tag value with HTML toolbar

You can also use the HTML toolbar for editing the tagged value of the element on a symbol in a diagram.

To use the HTML toolbar for editing an element tag value:

1. Click the Text tagged value on a Class shape (Figure 148 on page 381).
2. Then click it again. Tagged value will go into editing mode and the HTML toolbar will open (Figure 149 on page 381).

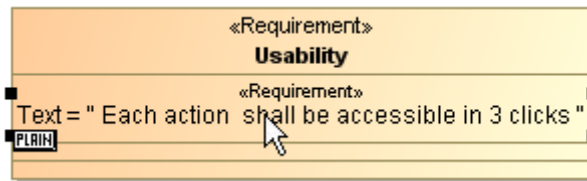


Figure 148 -- Selecting the Text Tagged Value on a Class Shape

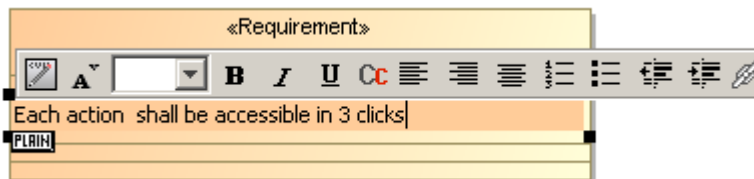


Figure 149 -- Editing the Text Tagged Value














HTML editor toolbar




HTML editor toolbar contains buttons to edit text (Figure 150 on page 382). See buttons description in Table 3 on page 383.



Figure 150 -- HTML toolbar

TABLE 3. Buttons of HTML toolbar

Button	Function
 Font properties	Select font style of the text.
 Font Size	Select font size of the text.
 Bold	Set text as bold.
 Italic	Set text as italic.
 Underline	Set text as underlined.
 Text Color	Select font color of the text.
 Align left	Align text to the left side border.
 Center	Center the selected text.
 Align Right	Align the selected text to the right side border.
 Insert Hyperlink	Add a hyperlink to the desired file, model element, or web page for the selected text. The Insert Hyperlink dialog box opens.
 Unordered list	Change text style to bullet list.
 Ordered list	Change text style to numbered list.
 Decrease Indent	Decrease indent by moving text closer to the left border.

Button	Function
 Increase Indent	Increase indent by moving text closer to the right border.
 Hyperlink	Add hyperlink to file, web page, element or symbol. The Edit Hyperlink dialog box opens. For more information, see “Adding a hyperlink to the model element ” on page 363.
 Advanced HTML Editor	Edit text with advanced HTML editor. The Advanced HTML Editor dialog box opens.



Advanced HTML editor

To open the **Advanced HTML Editor** dialog box (Figure 151 on page 385):

- Click the **Advanced HTML Editor** button in the HTML toolbar (Figure 150 on page 382).

In the **Advanced HTML Editor** dialog box you can change the text style, insert symbols, image, table and perform other actions using buttons.

Click the **HTML source** tab to view HTML source.

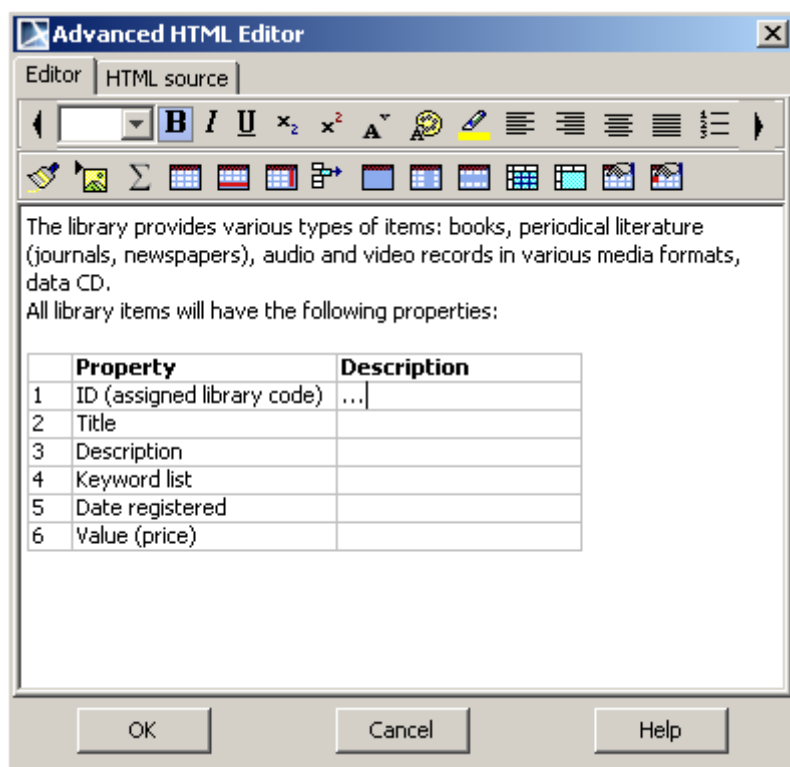


Figure 151 -- The Advanced HTML Editor dialog box



Copying/Opening Element URLs

You can now copy a project element URL to a clipboard and share it with other as a quick reference to model elements

To copy a project element URL, do any of the following:

- Select **Copy Element URL** from the element shortcut menu in the Containment tree to copy the URL to a model element.

or

- Select the element symbol in a diagram and click **Edit > Copy Element URL** on the main menu to copy the URL to element symbol.

You can open any elements through their URLs by clicking the **Open Element from URL** command and the element will be highlighted in the Containment tree or in the diagram. Custom URL "mdel://" is registered into windows registry. Activating the URL in other applications will allow you to start MagicDraw, open the project (if possible), and select any elements. You can paste URLs from the clipboard to any MagicDraw diagrams. Hyperlinks also can hold URLs of any model elements.

7 TOOLS

MagicDraw provides the following tools and wizards to help you quickly and easily perform design tasks.

- **"Model Merge"** - Model Merge enables porting changes between different project versions.
- **Report Wizard** - report engine built on top of the Velocity Engine. It is described in a separate document "MagicDraw ReportWizard UserGuide.pdf", which is located in the MagicDraw installation directory, Manual folder.
- **"Pattern Wizard"** - creates various GOF, Java, Junit, CORBA IDL, XML Schema and others design patterns.
- **"Creating Setters / Getters"** – creates getting and setting operators for attributes defined in the class.
- **"Implementing or Overriding Operations"** – creates defined operations down the inheritance tree.
- **"Model Transformation Wizard"** - is a data set, which fully describes the action of applying the model transformation on a set of packages. Transformation type, parameters, packages set to transform, and the destination of the model transformation results are the parts of model transformation.
- **"Resource Manager"** - MagicDraw Resource Manager functionality allows you to manage resources (Profiles, Plugins, Templates, Language resources, Case studies/ examples, Custom diagrams, and others).
- **"Spelling Checker"** - Spell Checker will check spelling as you type. Select what you want to be spell checked (the whole project or some specific parts).
- **"Import Data to MagicDraw"** - Data Import to MagicDraw using RConverter and data import from other tools to MagicDraw.

Model Merge

[View Online Demo](#) Model Merge

NOTE Model Merge functionality is available in Standard Edition and above for an additional fee.

Definitions

Name	Definition
Model Merge	Model Merge enables copying changes between different project versions. This functionality is usually needed when there are several branches that reflect different releases or versions of the product, e.g. when certain fixes have to be copied from a release branch to the main-stream development.
2-way merge	2-way merge is a simple merge, which compares two projects and joins them into one.
3-way merge	3-way merge does not only compare and merge two projects into one project, but it also considers the common ancestor of both projects.
Contributor	Participants of 2-way and 3-way merges are called contributors.
Ancestor	Ancestor is the common parent project for two projects.
Target, Source	In 3-way merge, the target is the contributor into which changes are copied, and the source is the contributor from which changes are copied.
Conflict	A conflict is two changes that are incompatible, i.e. changes that cannot be accepted together. For more information about conflicting changes, see “Conflicting changes” on page 426.

Introduction to Merging

Model merge enables copying changes between different project versions. This functionality is usually needed when there are several branches that reflect different releases or versions of the product, e.g. when certain fixes have to be copied from a release branch to the mainstream development.

Merge functionality in MagicDraw works both on projects stored in MagicDraw Teamwork Server or files.

1. Select **Project Merge** from the **Tools** main menu. The **Merge Projects** dialog box opens.
2. Select the source, target, and ancestor projects (changes are going to be copied from the source to the target project). The source and target projects can be stored in the system file or the Teamwork Server. If both projects are stored in the Teamwork Server, the ancestor is determined automatically. For more information about the **Optimize for** option, see “Controlling Merge memory usage” on page 465.
3. The **Merge** window appears, to let the user accept or reject the changes and resolve the conflicts that occurred in both contributors (e.g. when the same class is edited in both contributors).
4. Confirm the changes made to the target.

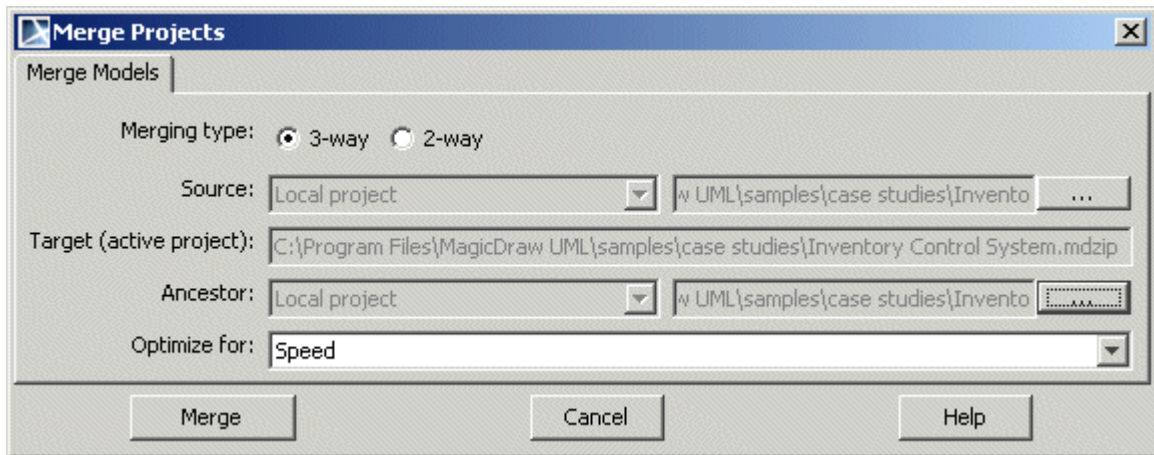


Figure 152 -- The **Merge Projects** dialog box

Two merge types are implemented:

- “3-way merge” on page 389.
- “2-way merge” on page 390.

3-way merge

Conceptually 3-way merge is a reconciliation of 2 difference sets. To merge projects v2 and v3, which have common ancestor v1, difference sets of projects v1-v2 and v1-v3 must be reconciled. 3-way merge can be used for merging changes from one branch into another.

See Figure 153 on page 390, where the merging changes from one branch into another is presented. Branch *b.1* is created from the project version *i*. Team members work in parallel on project version *i*

and make changes in project branch *b.1*. The merge is performed and changes made to branch are copied to the trunk.

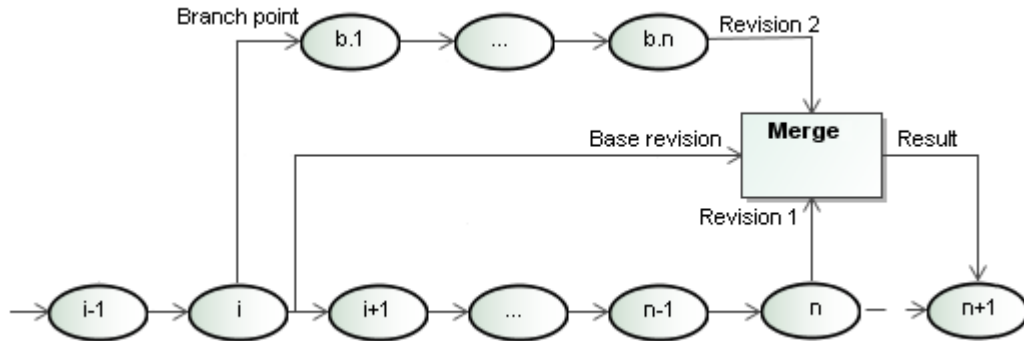


Figure 153 -- 3-way merge

2-way merge

2-way merging is a specific case of 3-way merging. This type of merging is usually used in a non-teamwork environment. In the 2-way merging there is no explicit ancestor available, thus the target version or file is taken as an ancestor.

See Figure 154 on page 391, where the first user works with project version *i* and creates project versions *i+1*, ..., *n-1*, *n* and the other user creates branch *b.1* of the project *i* and later creates project versions *b.1*, ..., *b.n*. Merging of the following projects is performed: project of the first user (*n*) and project of the second user (*b.n*) are merged.

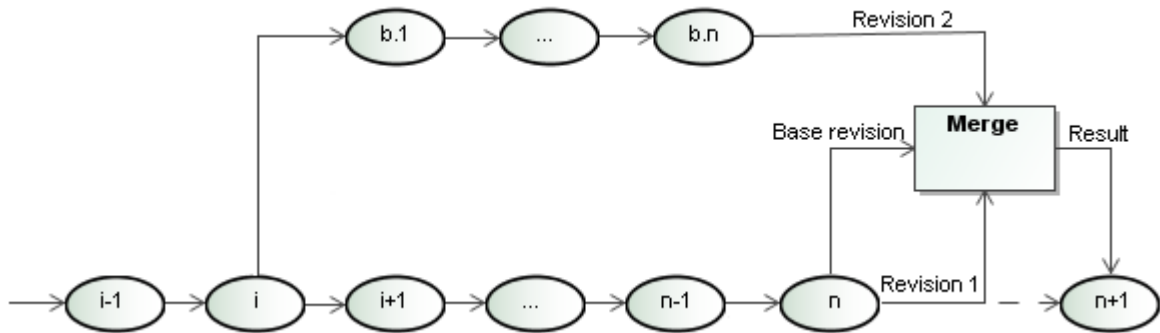


Figure 154 -- 2-way merge

Model Merge Concepts

A change is a delta between the ancestor and participants of a 2-way or 3-way merge. The participants of 2-way and 3-way merges are called contributors. Changes in Model Merge are classified as addition, deletion, modification, move, and order changes. Every change can be accepted or rejected and has dependent changes. If conflicts are detected, the user will be informed which change created the conflict.

Merging begins with building a composite change tree, which consists of model, diagramming, and non-model changes.

Model, diagramming, and non-model changes depend on each other. This means that accepting or rejecting one change not only will accept or reject other changes in the same change tree, but also in other change trees.

For more information about model merge concept, see "Merging concepts in details" on page 423.

Introductory Case Studies

This section will present the following case studies:

- "Case Study 1 - 3-way Merge and Analysis" on page 392.
- "Case Study 2 - merging in Teamwork System" on page 398.

- “Case study 3 - Copying changes from branch to Trunk (in Teamwork) with conflict resolution” on page 408.

These case studies will show how merging is performed by accepting or rejecting changes and resolving conflicts.

Case Study 1 - 3-way Merge and Analysis

This case study presents step-by-step instructions for merging projects and analyzing results.

Merge functionality in MagicDraw works for both on projects stored in MagicDraw Teamwork Server and files. This case study presents projects. For information about how merging is performed in Teamwork, see “Case Study 2 - merging in Teamwork System” on page 398.

Inventory Control System.mdzip project will be used to show basic merge functionality. You can find this project in the *MagicDraw installation folder > samples > case studies* folder.

In this case study User1 and User2 will make changes in parallel in the same project and later User1 will copy the changes from User2 file to his/her own file.

Actions before merging

User1 opens the *Inventory Control System.mdzip* project and makes the following changes to the project.

1. Create a new *Customer Group* class in the *Top Level class diagram* class diagram, in the *Static View* package (see Figure 155 on page 393).
2. Save the project.

User2 opens the original *Inventory Control System.mdzip* project and makes the following changes to the project.

1. Rename the *Shipment* class to *Delivery* in the *Product Shipment object diagram* class diagram (see Figure 156 on page 394).
2. Save this project.

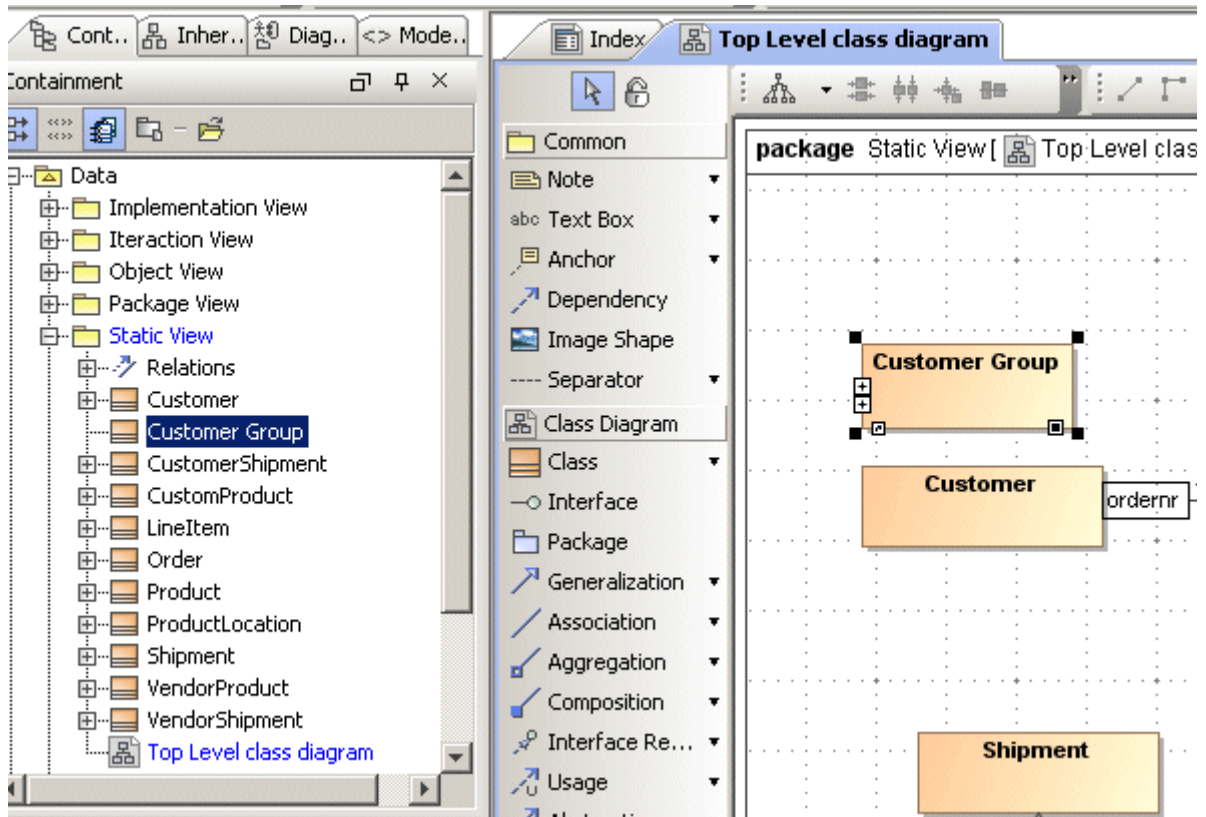


Figure 155 -- Changes made by User1 in the Inventory Control System.mdzip project

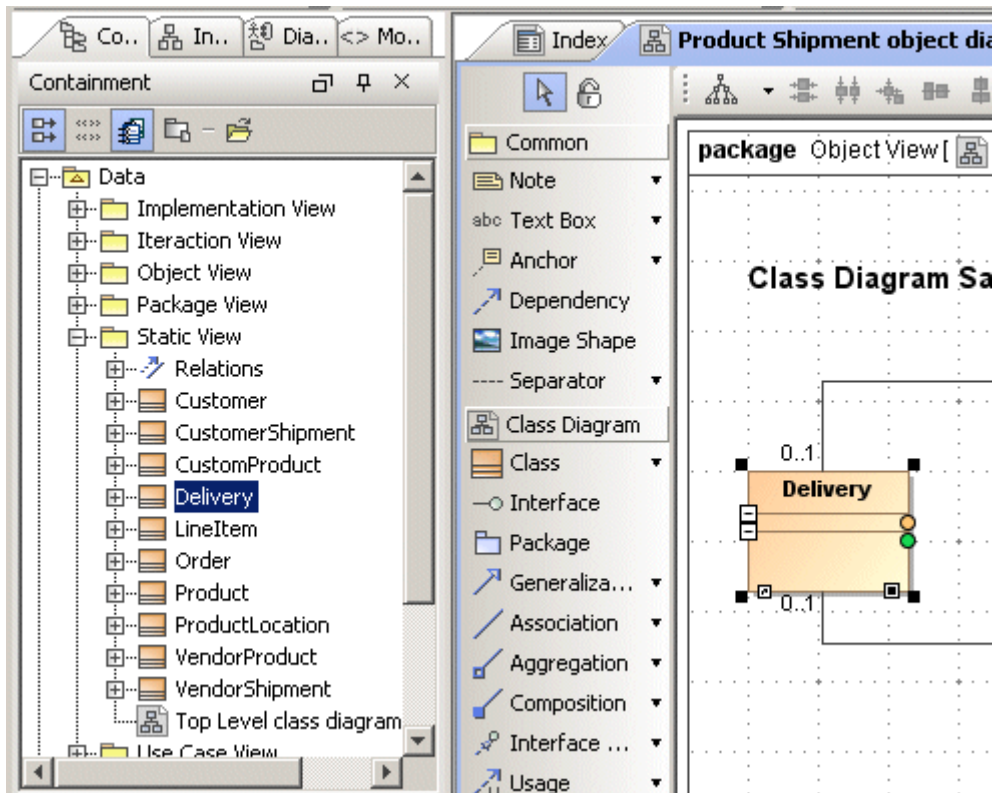


Figure 156 -- Changes made by User2 in the Inventory Control System.mdzip project

User1 merges the projects

1. User1 opens the target project - the *Inventory Control System.mdzip* project, which includes changes made by himself/herself.
2. Chooses the projects to merge and starts merging:
 - 3 From the **Tools** menu, select **Project Merge**. The **Merge Projects** dialog box opens (see Figure 157 on page 395).
 - 4 Click the "..." button in the **Source** group and select the *Inventory Control System.mdzip* project, which includes the changes made by User2.
 - 5 Click the "..." button in the **Ancestor** group and selects the original *Inventory Control System.mdzip* project that contains no changes.

- 6 Click the **Merge** button. The **Merge** window opens (see Figure 158 on page 396).

NOTE

A 3-way merging needs an ancestor project. If there is no ancestor project, use 2-way merge. For more information about 2-way and 3-way merge, see the “Introduction to Merging” on page 388.

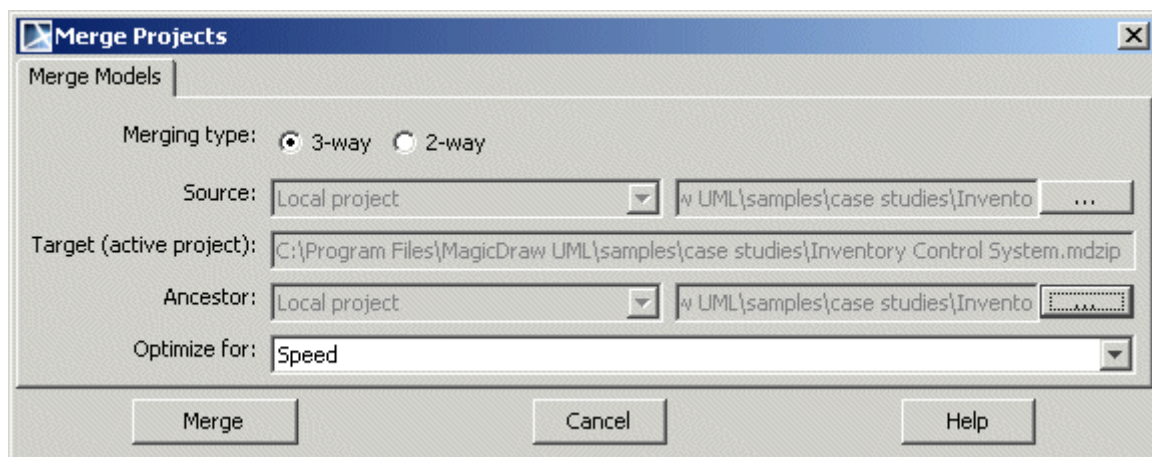


Figure 157 -- The Merge Projects dialog box

Analyzing merging results

The **Merge** window allows reviewing differences. The merger automatically applies changes made in both source and target projects and highlights those changes in the Merged Result tree (see Figure 158 on page 396).

Grey lines underneath the Static View means that there are changes inside the package. Expand the *Static View* node and see that *Customer* Group class is added in the package, the *Shipment* class and the *Top Level class diagram* are modified (see Figure 159 on page 397).

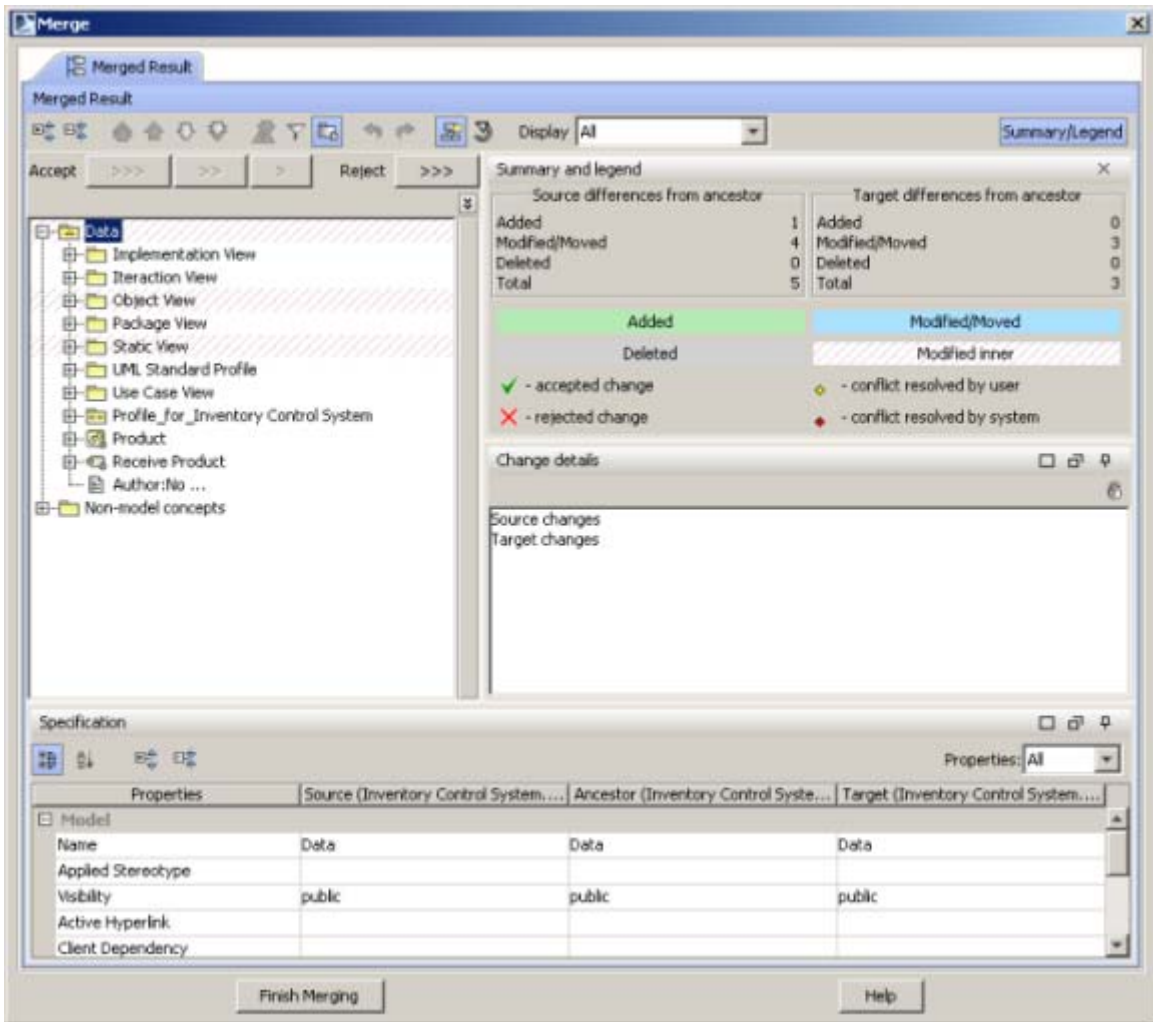


Figure 158 -- The Merge window - Merge Result tree

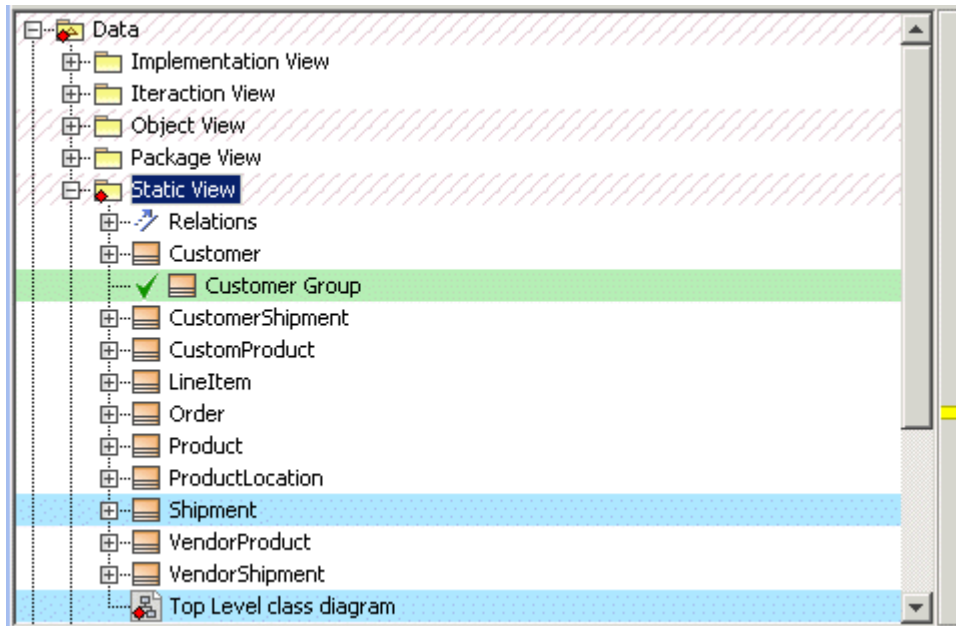


Figure 159 -- The Merged Result tree - the *Customer Group* class is added in the *Static View* package

Completing the merging procedure

Confirm the changes made to the target by pressing the **Finish Merging** button. The changes in the source project will be copied to the target project, in the example the *Customer Group* class is added and the *Shipment* class is renamed as *Delivery* (see Figure 160 on page 398).

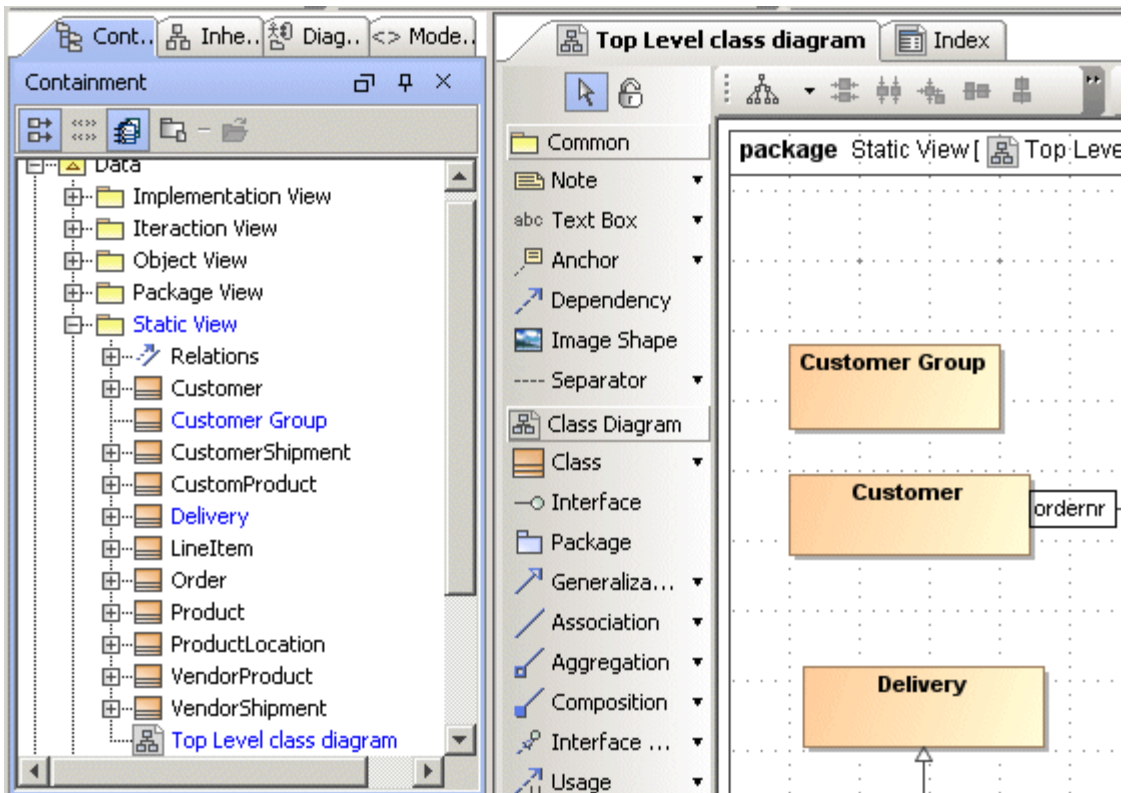


Figure 160 -- The Inventory Control System.mdzip project after merge

Case Study 2 - merging in Teamwork System

This case study shows how simple changes can be copied from the release branch to the main development branch when *User1* and *User2* are working on the same project in Teamwork Environment.

User1 and User2 actions before merging projects in the Teamwork System

User1 adds the *Inventory Control System.mdzip* project to the Teamwork Server.

User2 creates a branch for the *Inventory Control System.mdzip* project and modifies it.

1. To create a project branch:

- 2 From the **Teamwork** menu, select **Projects**. The **Edit Projects** dialog box opens (see Figure 161 on page 400).
- 3 Select the *Inventory Control System* project and click the **Versions** button. The **Versions** dialog box opens (see Figure 162 on page 401).
- 4 Select the first version of the project and click **Create Branch**. Now *User2* has created a branch, which is derived from the project version 1 from trunk (see Figure 163 on page 402).
5. To modify a project branch.
 - 6 In the **Versions** dialog box, select *Release 1.0* and click the **Open** button to open the project. The *Release 1.0* branch of the *Inventory Control System* project opens.
 - 7 Create the *Customer Group* class in *Top Level class diagram*, in the *Static View* package (see Figure 164 on page 403).
 - 8 Commit changes.

For more information about branching in Teamwork, see Project branching in Teamwork section, *MagicDraw Teamwork UserGuide.pdf*.

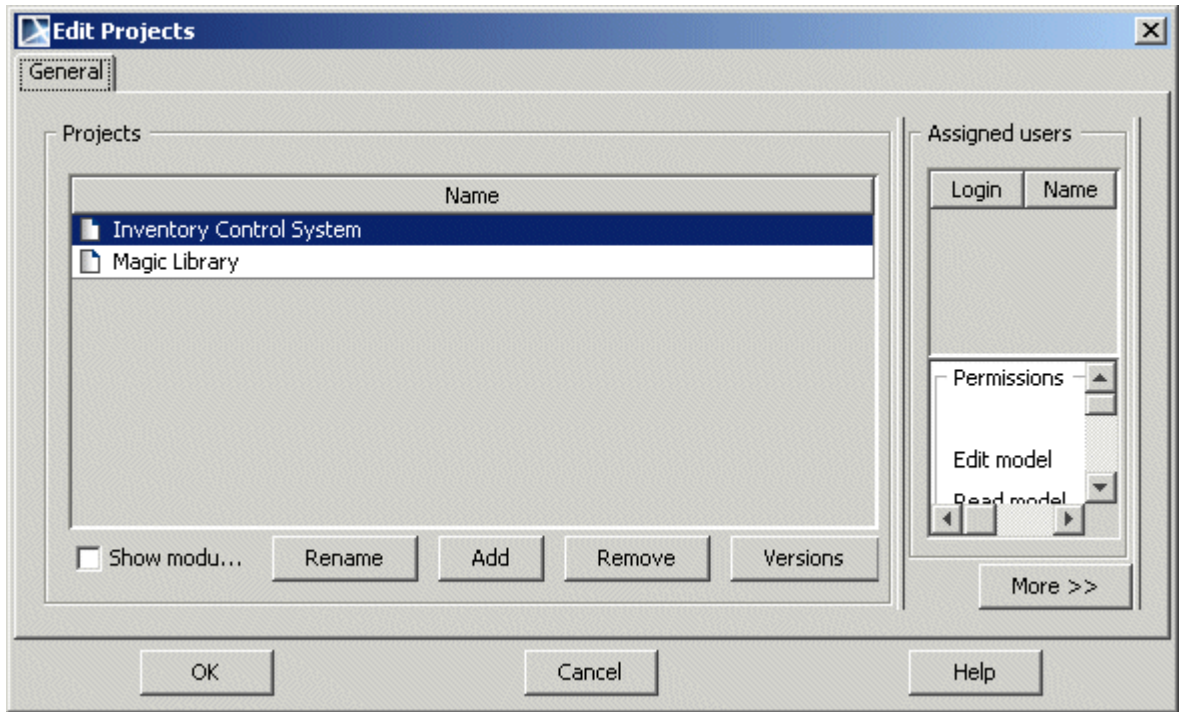


Figure 161 -- The Edit Projects dialog box

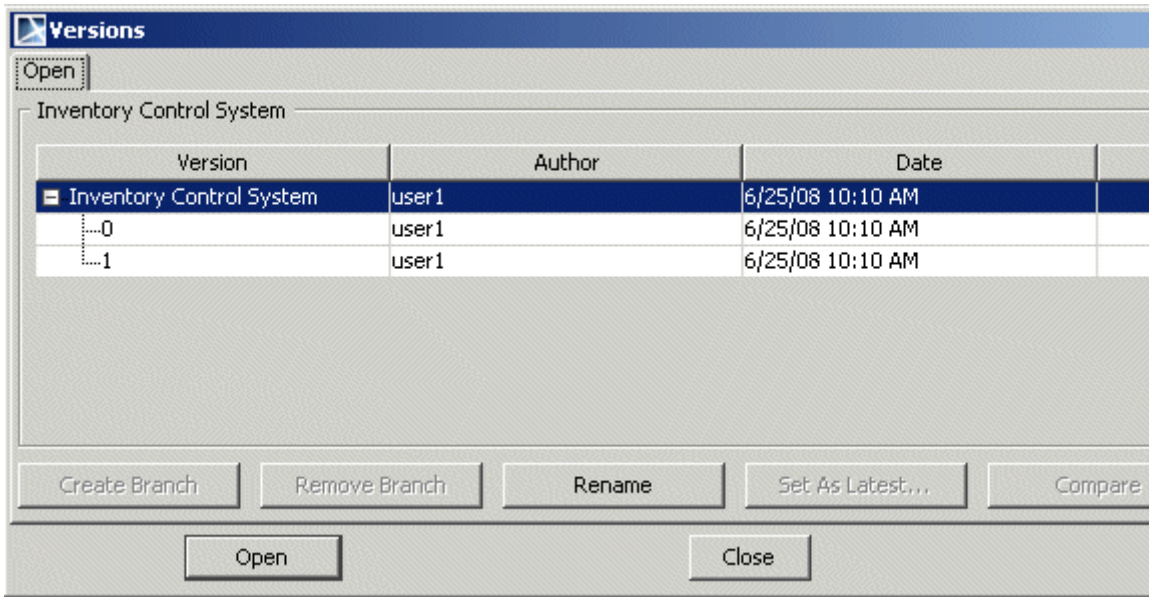


Figure 162 -- The Versions dialog box

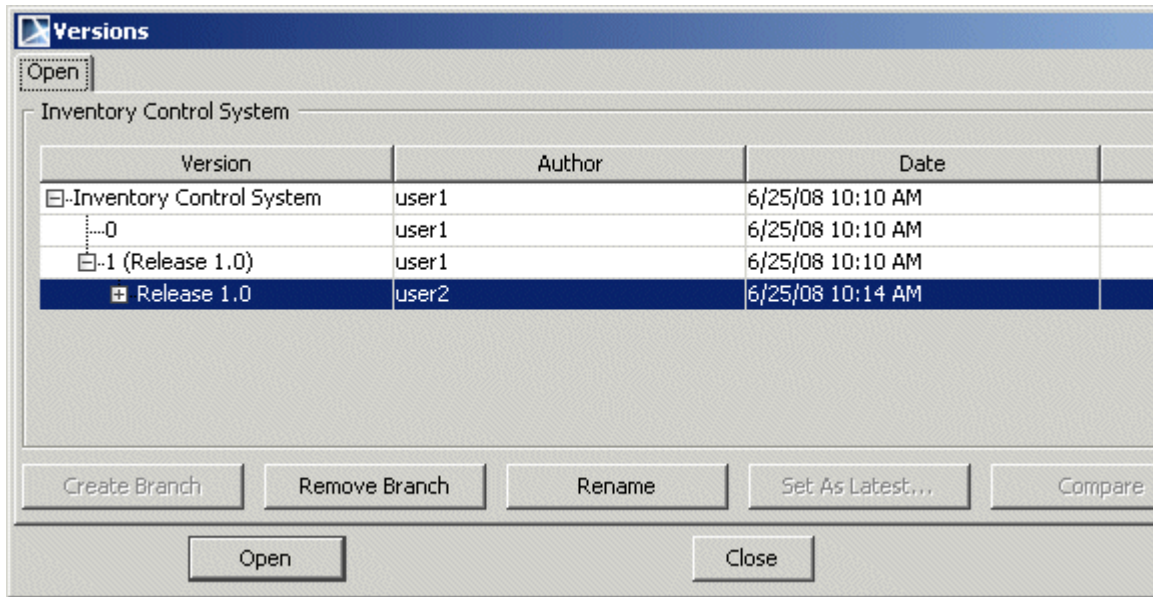


Figure 163 -- The Inventory Control System project versions with created Release 1.0 branch

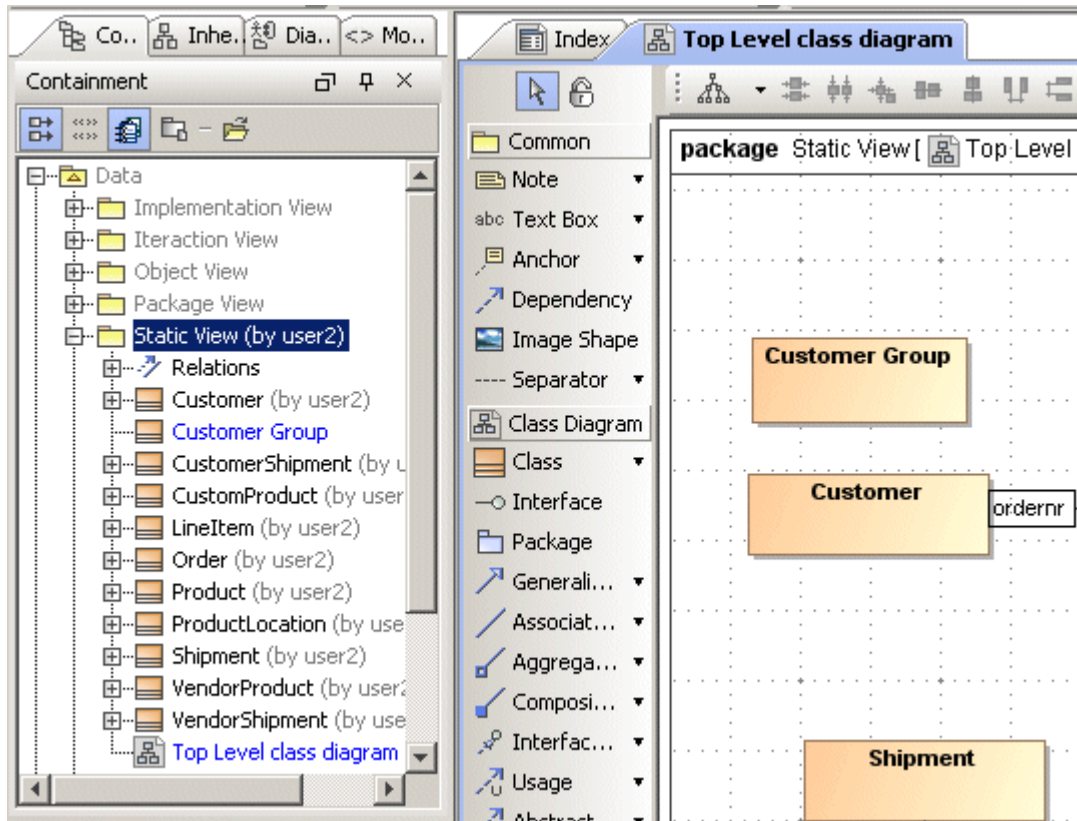


Figure 164 -- Changes made by User2

User1 actions to merge changes from branch into the trunk

1. To merge changes from branch into the trunk by opening the latest version of trunk.
2. From the **Teamwork** menu, select **Open Teamwork Project**. The **Open Teamwork Project** dialog opens (see Figure 165 on page 404).
3. Select the *Inventory Control System* project and click **Open**. The latest version of trunk is opened.
4. To merge changes from the latest version of the project branch to the current active project.
5. From the **Teamwork** menu, select **Merge From**. The **Select Teamwork Project** dialog opens (see Figure 166 on page 405).

- 6 Expand the 1st node and select the *Release 1.0* branch (see Figure 167 on page 405).
7. Click the **Merge** button. The **Merge** window opens (see Figure 168 on page 406). The **Merge** window allows the user to review differences and complete the merge procedure.

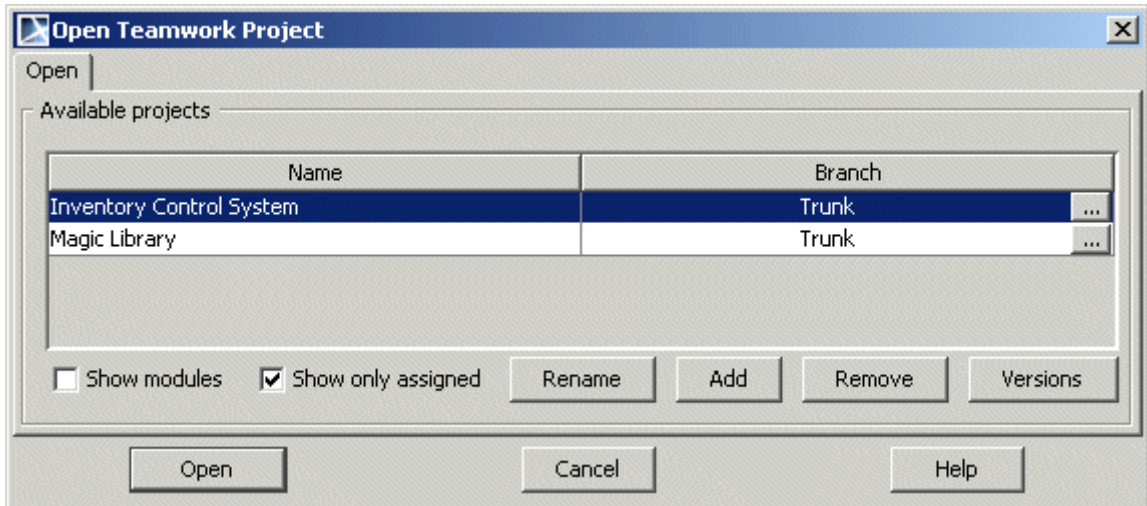


Figure 165 -- The Open Teamwork Project dialog box

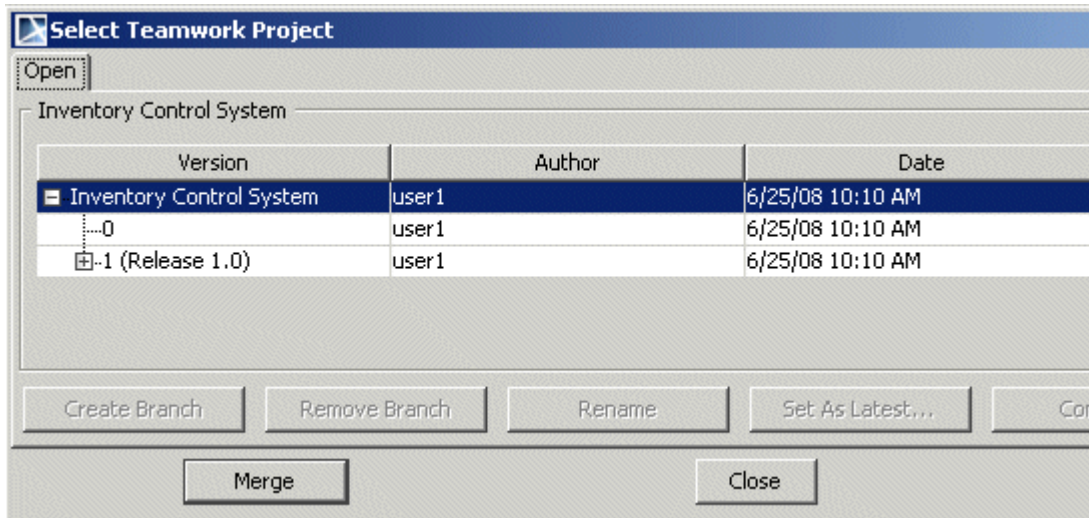


Figure 166 -- The Select Teamwork Project dialog box

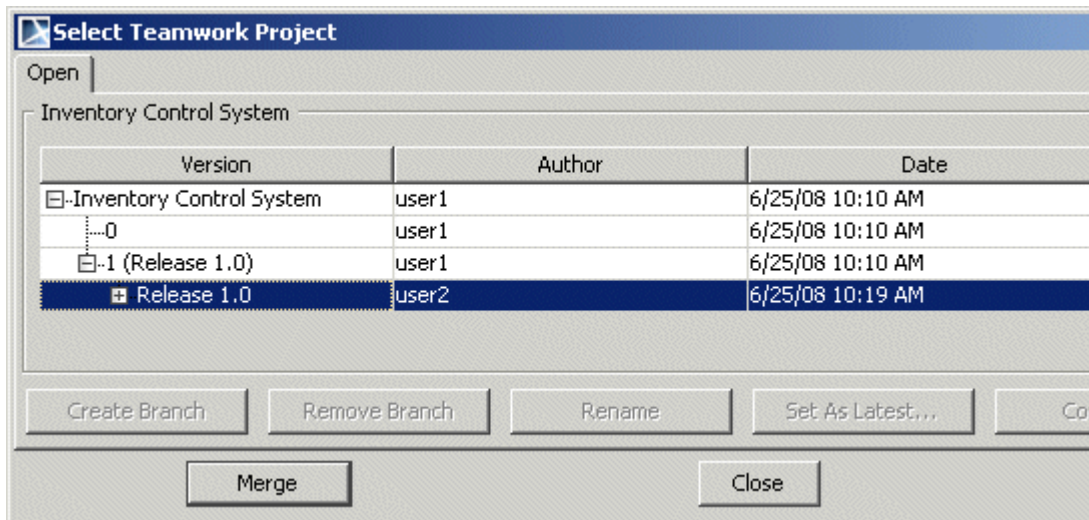


Figure 167 -- The Select Teamwork Project dialog box with selected branch from which we are going to copy changes

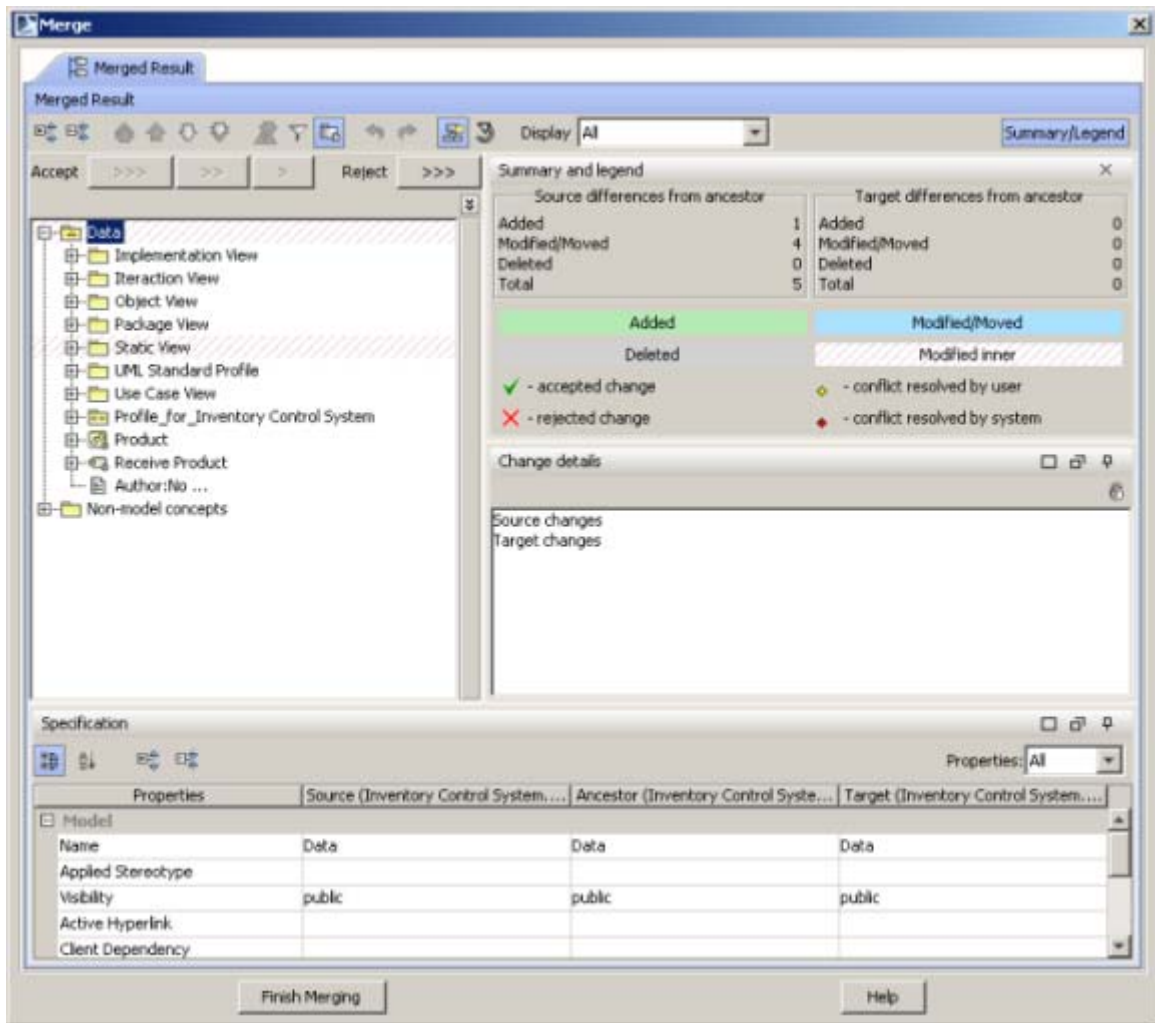


Figure 168 -- The Merge window

Analyzing the merge results

Expand the *Static View* node in the Merge Result tree and you can see that *Customer Group* class was added to the package and the *Top Level class diagram* has modification changes (see Figure 169 on page 407).

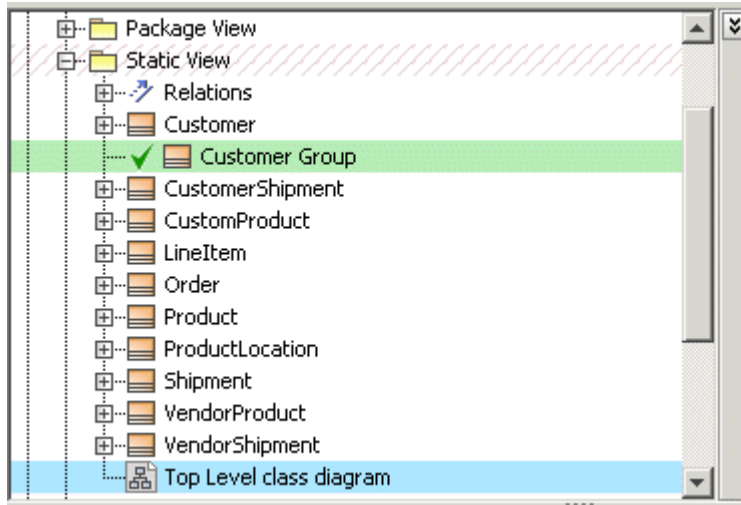


Figure 169 -- Case study 2 - the Merge window, Merge Result tree

User1 confirms merging changes

1. User1 confirms changes made to the target by clicking the **Finish Merging** button in the **Merge** window. The changes from the source project are copied to the target project, in this sample the *Customer Group* class is added to the trunk project (see Figure 170 on page 408).
2. User1 commits the changes to the Teamwork Server.

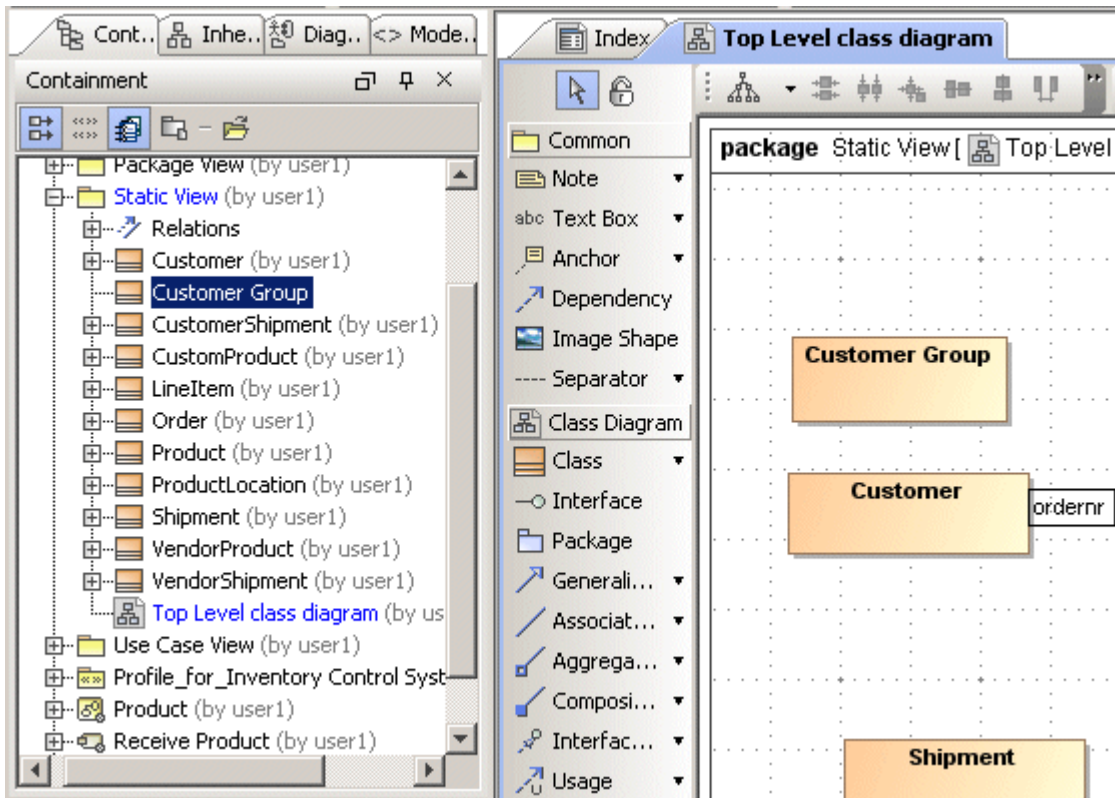


Figure 170 -- Changes are copied from the branch to the Trunk - the Customer Group class is added to the Trunk project

For more information about analyzing merging results, see “Analyzing Merging Results” on page 429.

Case study 3 - Copying changes from branch to Trunk (in Teamwork) with conflict resolution

This case study analyzes how conflicts are resolved during the merging process.

Actions Before Merging

User1 adds the *Inventory Control System.mdzip* project to the Teamwork Server and modifies it.

1. Rename the *Customer* class as *Client* class (see Figure 171 on page 409). The *Customer* class is located on *Top Level class diagram*, in the *Static View* package.
2. Commit changes to the Teamwork Server.

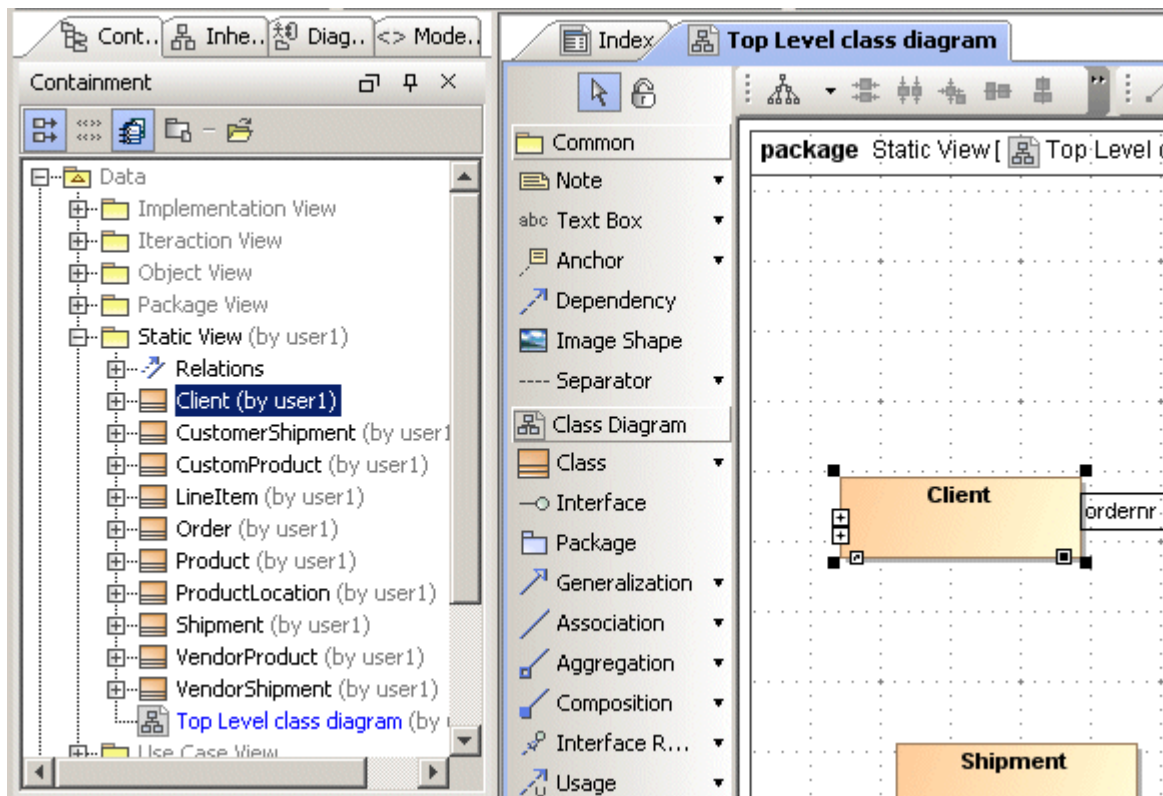


Figure 171 -- Case Study 3 - User1 changes in the project

User2 creates a branch of the *Inventory Control System* project and modifies it.

1. To create a project branch:
 - 2 From the **Teamwork** menu, select **Projects**. The **Edit Projects** dialog box opens (see Figure 172 on page 410).
 - 3 Select the *Inventory Control System* project and click the **Versions** button. The **Versions** dialog box opens (see Figure 173 on page 411).
 - 4 Select the first version of the project and click **Create Branch**. Now User2 has created a branch, which is derived from the project version 1 from trunk (see Figure 174 on page 412).

5. To modify a project branch.
- 6 In the **Versions** dialog box, select *Release 1.0* and click the **Open** button to open the project. The *Release 1.0* branch of the *Inventory Control System* project opens.
- 7 Rename the *Customer* class as *Customer Group* class (see Figure 175 on page 413). The *Customer* class is located on *Top Level class diagram*, in the *Static View* package.
- 8 Commit changes.

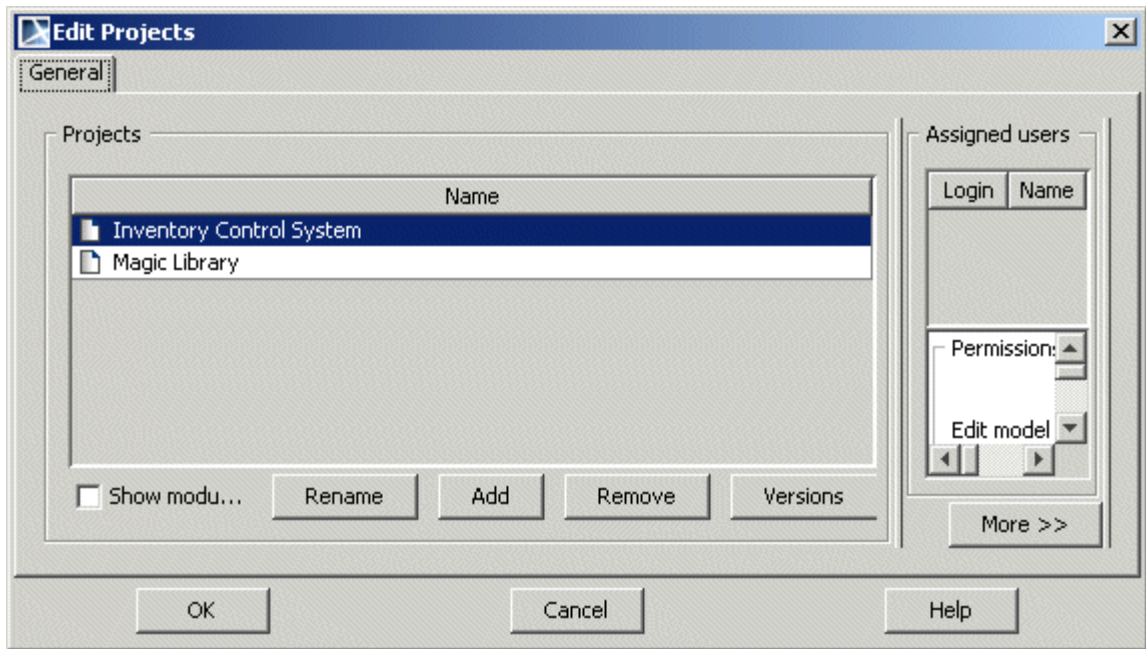


Figure 172 -- Case Study 3 - the Edit Projects dialog box

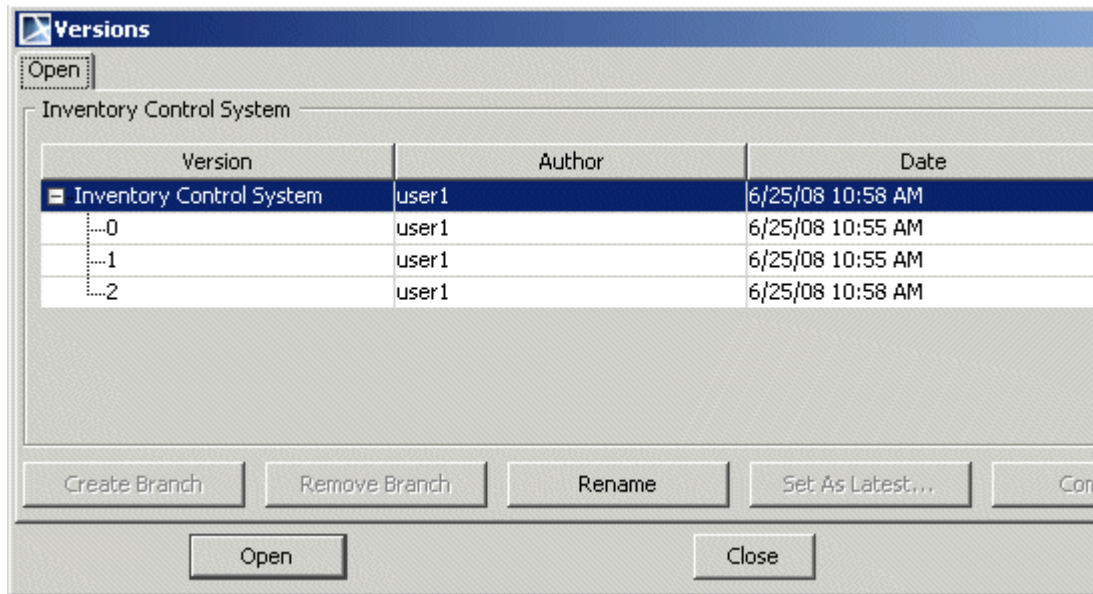


Figure 173 -- Case Study 3 - the Versions dialog box

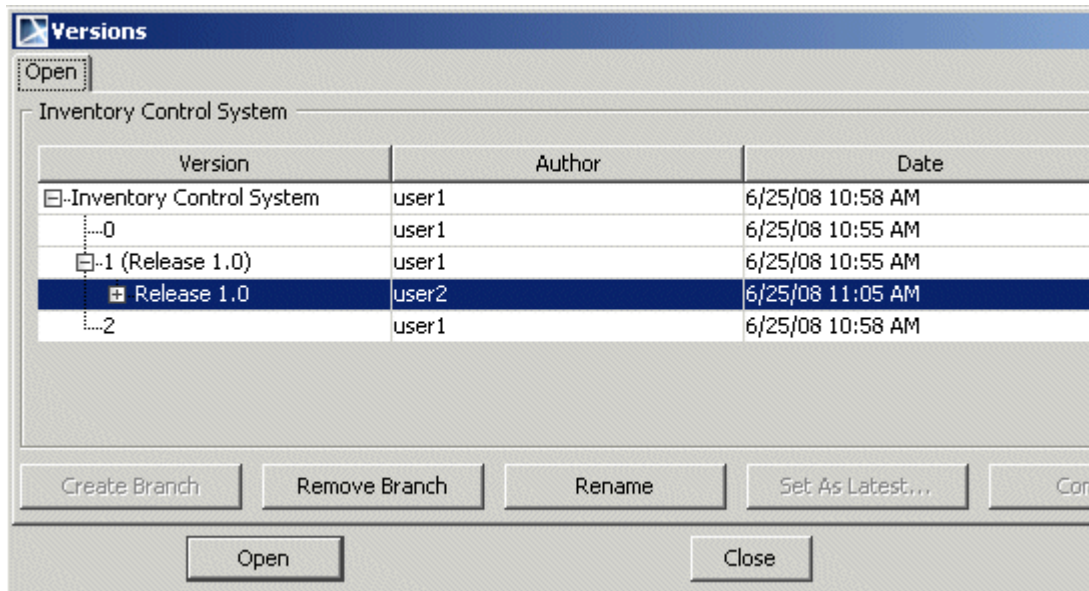


Figure 174 -- Case Study 3 - the branch Release 1.0 is created

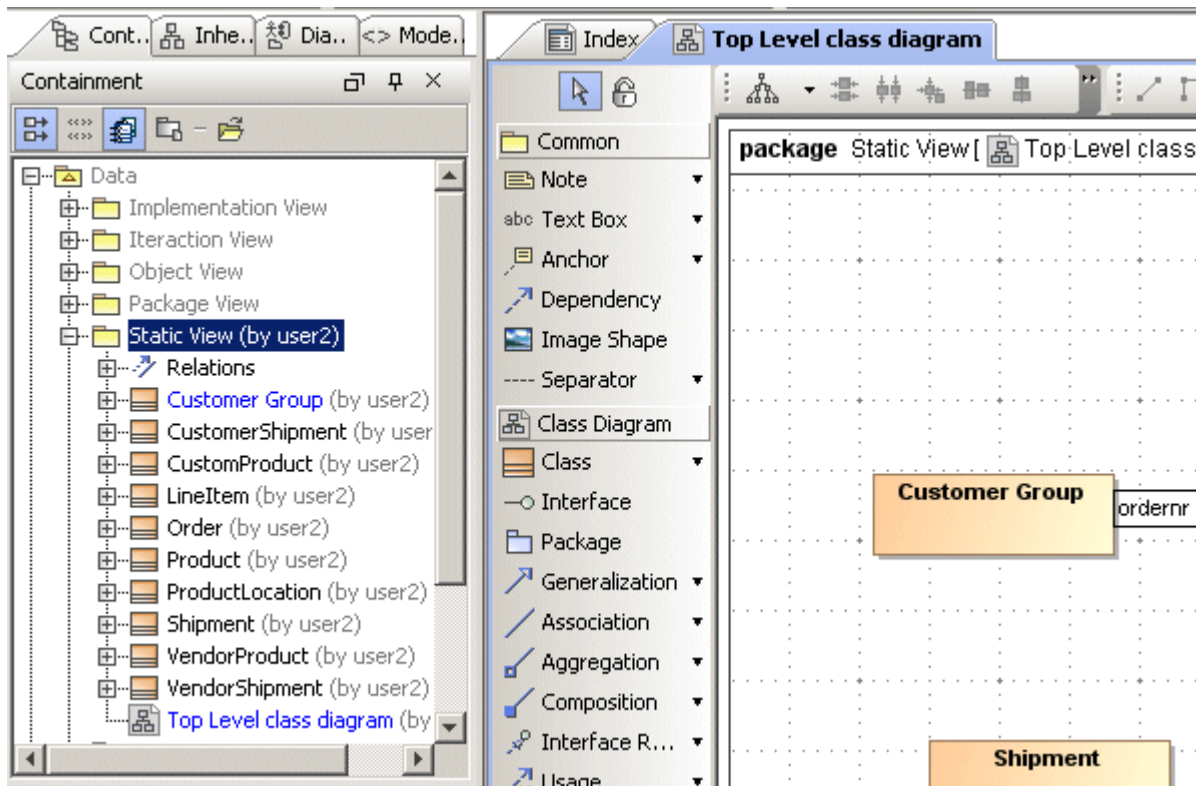


Figure 175 -- Case Study 3 - User2 changes made to the project

User1 merges changes from branch into trunk

1. To merge changes from branch into the trunk by opening the latest version of the trunk.
2. From the **Teamwork** menu, *User1* selects **Open Teamwork Project**. The **Open Teamwork Project** dialog opens.
3. Select the *Inventory Control System* project and click **Open**. The latest version of trunk opens.
4. To merge changes from the latest version of the project branch to the current active project.
5. From the **Teamwork** menu, choose **Merge From**. The **Select Teamwork Project** dialog opens.

6 Expand the 1 node and selects the *Release 1.0* branch (see Figure 176 on page 414).

7. Clicks the **Merge** button. The **Merge** window opens (see Figure 177 on page 415).

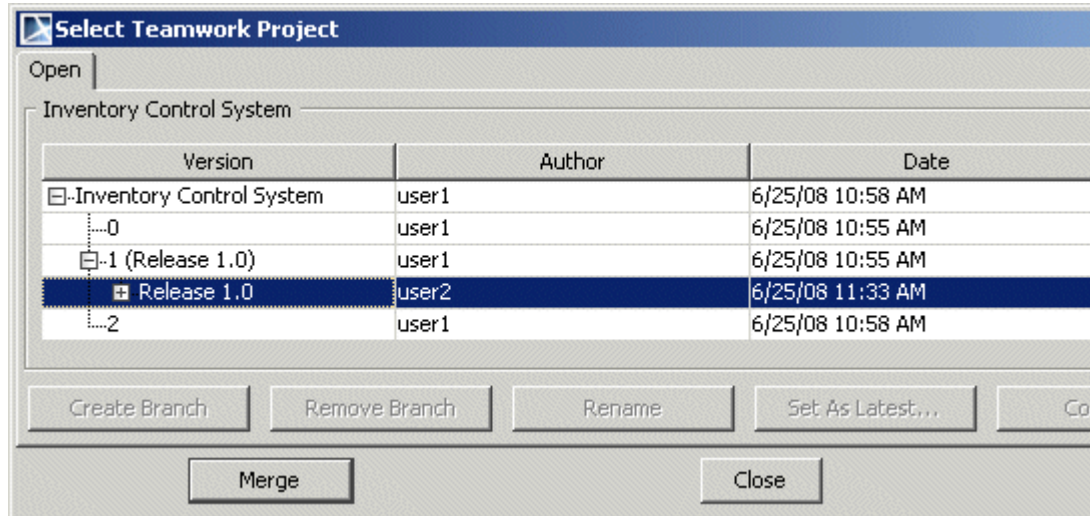


Figure 176 -- Case Study 3 - the Select Teamwork Project dialog box

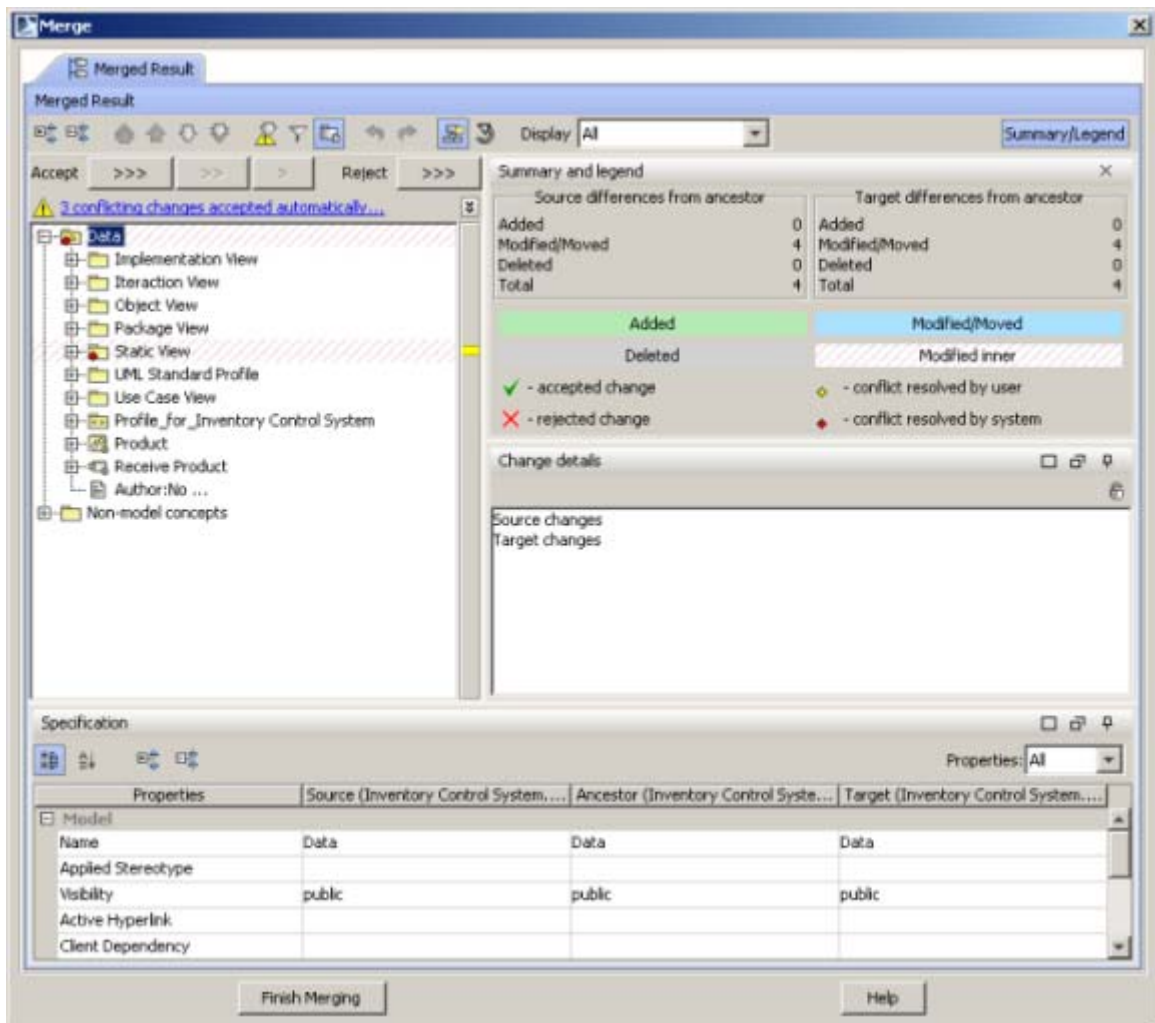


Figure 177 -- Case Study 3 - the Merge window

Analyzing conflicting changes

1. Figure 177 on page 415 shows the Merge Results tree in which there are conflicting changes that cannot be accepted. They are distinguished by red diamond shapes. The *Data* package has inner changes and conflicts and the *Customer* class has con-

flicts (the name was changed in the source and target projects) (see Figure 178 on page 417).

2. The merge engine automatically accepts all non-conflicting changes from the source and the target projects. Then it accepts all conflicting changes from the target project and rejects all the conflicting changes from the source. The user can review automatically accepted conflicting changes from the target and select to apply change from the other contributor.
3. To navigate the automatically accepted conflicting changes from the target project
 - 4 Click the **Go To Next Conflict** button in the quick navigation toolbar, to navigate the conflicting changes (see Figure 179 on page 418).
 - 5 The element that has conflicts (the *Customer* class) is selected (see Figure 180 on page 419). Its specification is also opened. The name of the *Customer* class is changed in the source and in the target project. Only the change in the target is automatically accepted.

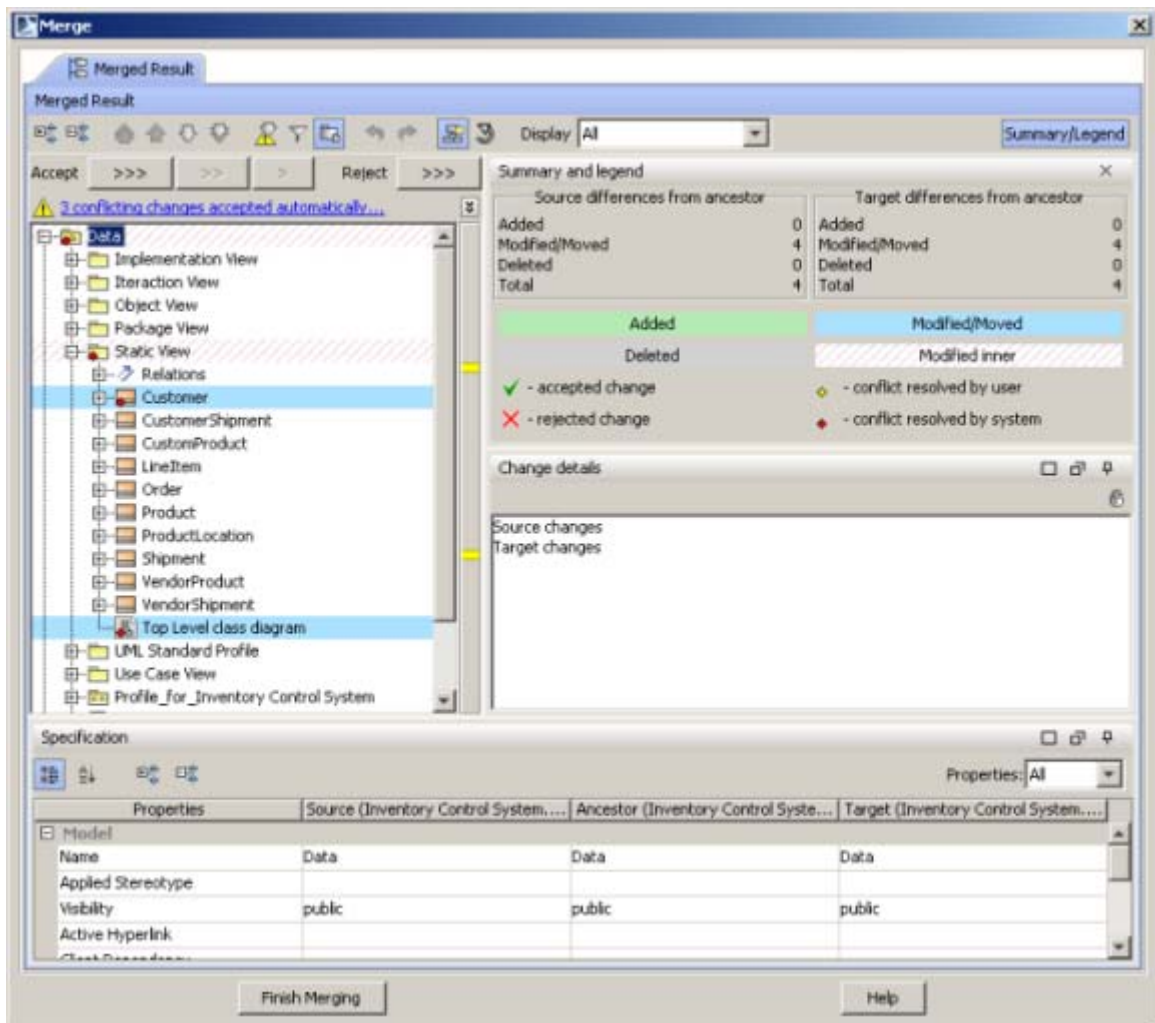


Figure 178 -- The conflicting change

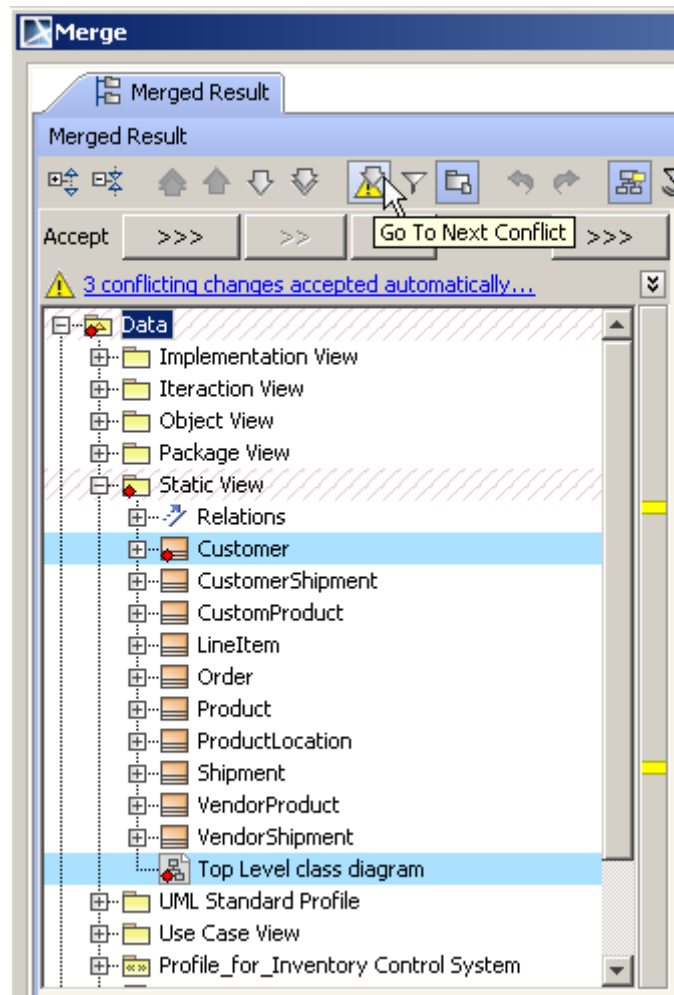


Figure 179 -- Case Study 3 - the Merge window, the Go To Next Conflict button

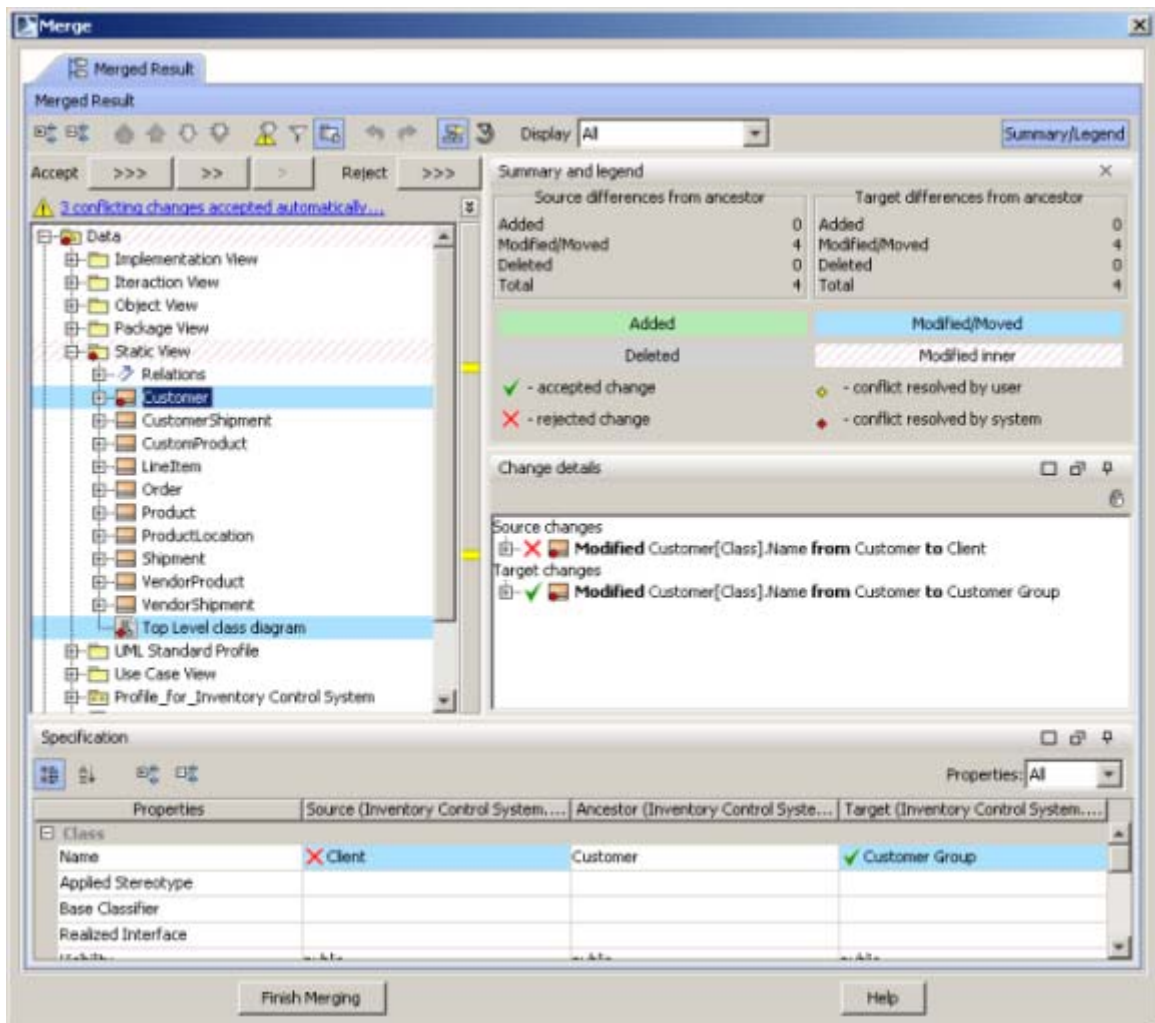


Figure 180 -- Case study 3 - the Merge window, Customer class is selected in the Merged Results tree

Changing the conflict resolution and finishing the merge

1. Let's change the automatic acceptance setting of the conflict resolution and accept the name change from the source project.

- 2 In the **Change details** panel of the *Customer* class, right-click on the not accepted change. The shortcut menu opens (see Figure 181 on page 420).
- 3 Accept the class name modification from the source project by selecting the **Accept** shortcut menu item. The name modification from the source project is accepted.
4. Click the **Finish Merging** button and commit changes to Teamwork.

User1 has successfully resolved conflicting changes and merged them into the target project. The *Customer* class is renamed to *Customer Group* (see Figure 182 on page 421).

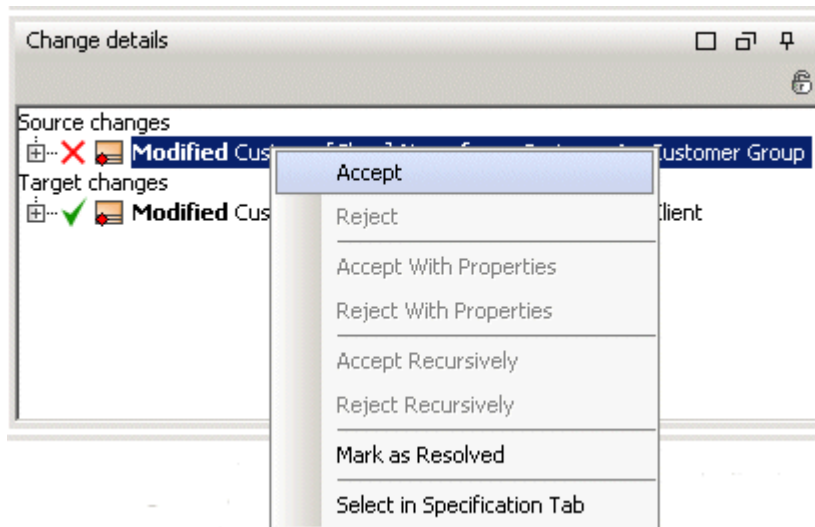


Figure 181 -- The Merge window, Specification panel with invoked shortcut menu

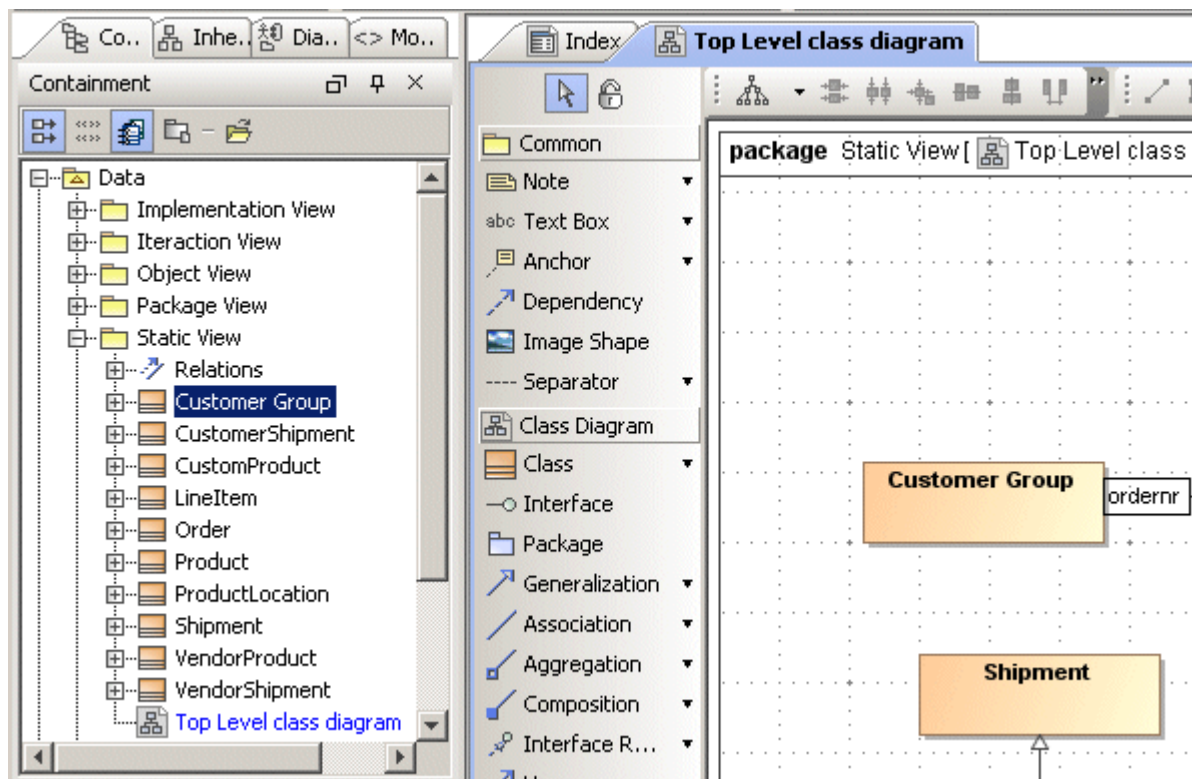


Figure 182 -- Project after merge

Case Study 4 - Representing DSL elements in the Merged results

Model elements and their properties are displayed and merged using DSL rules, specified as MagicDraw DSL customization artifacts.

This case study represents how DoDAF model elements are represented in the **Merge** window.

Actions before merging

In the DoDAF sample.mdzip project, which is located in the <MagicDraw installation folder>\samples directory, in the source project user makes the following change:

In the *OV-4 Organizational Structure* diagram, in the *Rear Production Division Target Materials Branch* element **Specification** dialog box, change the **Military Service Type** property to the **Army** value. Merge projects.

Analyzing the merge results

See the **Merge** window with the merge results in Figure 183 on page 423.

You can see that DSL elements in the Merge window are represented in the same way as they are represented in MagicDraw languages. That means that representation in the **Merge** window matches the assigned metaclasses in project:

- In the **Merged Results** tree, elements are displayed with DSL icons. In this example, see the icon of the **Organization** element, **Role** element, **OV-4 Organizational Structure** diagram.
- In the **Specification** pane, title of the element matches title of the DSL element. In this example, you can see that the Organization element specification properties are represented.
- In the **Specification** pane, DSL properties titles are represented. In this example, see the **Military Service Type** property.

In the **Change** details pane, information is presented using the titles of the DSL elements.

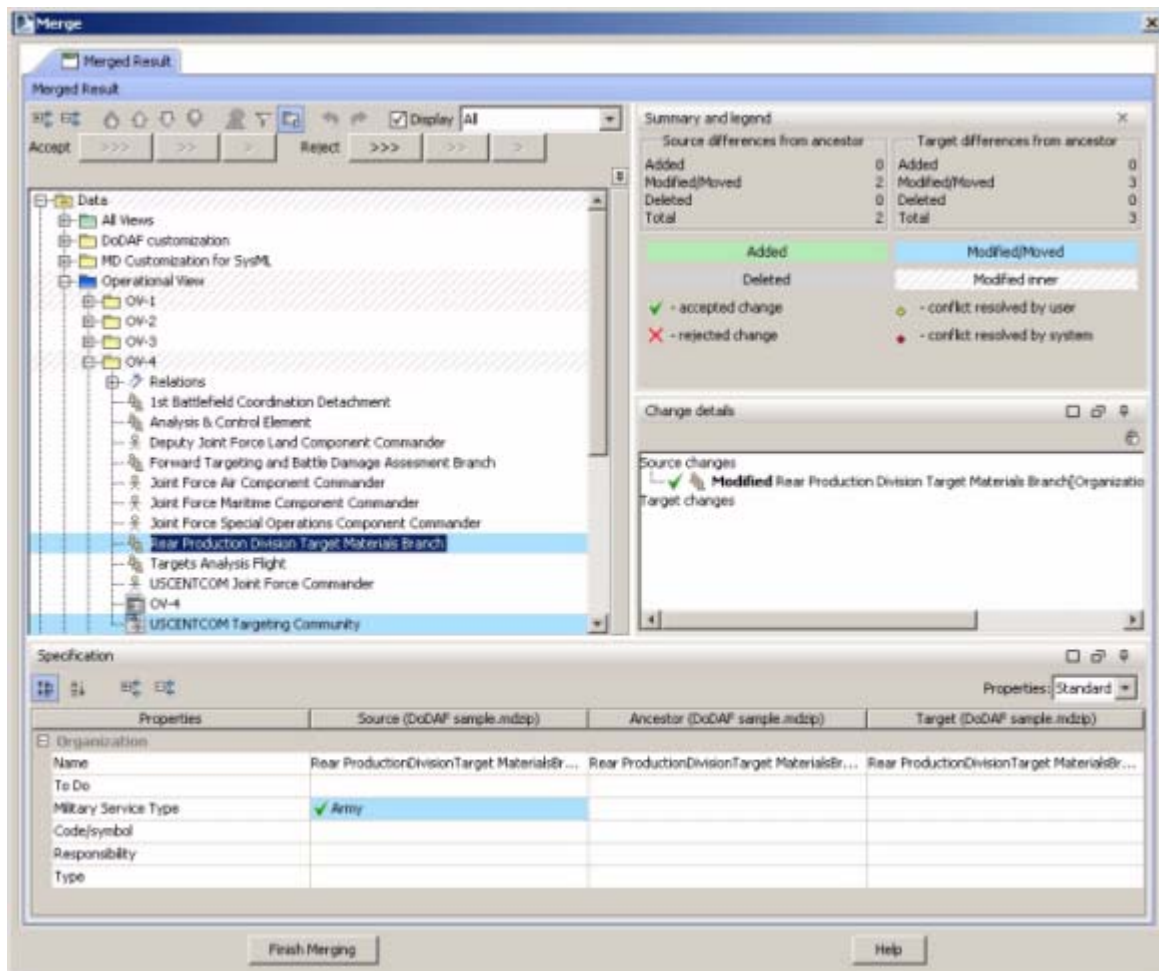


Figure 183 -- DoDAF elements representation in the Merge window

Merging concepts in details

This section will introduce: the change concept, change the type to accept and reject changes and conflict concept.

A change is a difference, found between the ancestor and participant of the 2-way or 3-way merge. For example, a merger compares elements of the ancestor project to those of the source or target project. Changes are classified as follows:

- addition
- deletion
- modification
- order

For a brief information about changes, see “Change types” on page 424.

Each change can be accepted or rejected and have dependent changes. See “Accepting or Rejecting changes” on page 425 for the details. Dependent changes are described in “Dependent changes” on page 425.

Changes can conflict with each other. See “Conflicting changes” on page 426.

Change types

Addition change

If an element is added to the contributor then an addition change occurs. If the addition change is accepted a new element will be created.

See the example of addition change in the “Addition changes” on page 438.

Modification change

If an element property in the contributor is modified then a modification change occurs (Note: element property means the element specification property, such as *Is Abstract*, *Multiplicity*, and others). For example, if the *Is Abstract* class property in the ancestor has a default value (which is false) and the *Is Abstract* property in the contributor is changed to true, a modification change would occur.

There are three types of modification changes:

- Addition modification change adds a value to the property.
- Deletion modification change removes a value from a property.

- Replacement modification change replaces one value with another. This type of modification change occurs only for properties that have multiplicity less or equal to 1.

For more information about modification change, see “Elements with modified properties” on page 441.

Deletion change

If an element is removed from the contributor, then a deletion change occurs in the removed element. For more information about deletion change, see “Deletion changes” on page 439.

Change order

If the order of elements in the ancestor and contributor differs, then an order change occurs.

For more information about order change, see “Order changes” on page 441.

Accepting or Rejecting changes

Every change, whether it is addition, modification, deletion, or order change, can be accepted or rejected.

Accepted changes are incorporated into the final project. Alternatively, they can be rejected and will not be applied to into the target project.

For more information about accepted or rejected changes in the **Merge** window, see “Analyzing Merging Results” on page 429.

Dependent changes

In some cases, other changes have to be accepted or rejected before accepting or rejecting the selected change. In other words, the selected change sometimes depends on other changes, then it is called a dependent change or dependant.

An example of a dependent change is a type change. If a class attribute type is changed to a type that is created by another change, then the attribute type change is dependent on the change that created the type. This means that type creation has to be accepted before accepting type modification change.

Another example is the type deletion, modification, and addition. Suppose there is an attribute type change in a contributor. The old type is deleted and a new type is added to the contributor. In this case, three changes are formed:

- deletion change (for the old type),
- addition change (for the new type),
- and modification change (for the property type).

These are also ownership changes, but they are accepted together with deletion and addition changes.

A modification change depends on an addition change and a deletion change depends on a modification change. Accepting the addition change does not mean accepting any other changes in this case. Accepting the modification change means accepting the addition change and then accepting itself. Accepting the deletion change means accepting the addition change, the modification change and the deletion change itself. The change dependencies for the described case is depicted in Figure 184 on page 426.

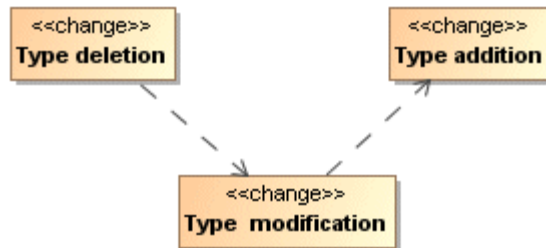


Figure 184 -- Sample of the dependent changes - type deletion, modification, and addition

Conflicting changes

A conflict is two differences that are incompatible with each other, i.e. changes that can not be accepted together. Every change can have several conflicting changes.

NOTE Conflicting changes occur only in a 3-way merge.

Some examples of the conflicting changes:

- Each contributor changed a class' name or any other element properties (metaproperty).

- One contributor added an operation to a class and the other contributor deleted the class.
- One contributor moved a class into one package while the other contributor moved it to another package.

For more information about the representation of the conflicting changes, see “Analyzing Merging Results” on page 429.

Building change tree

There are two groups of elements that are compared and merged:

1. Model/diagram elements
2. Non-model elements.

The Model/diagram tree is represented in the Merged Result tree as it is done in the Browser in the main MagicDraw window.

The Non-model tree is represented in the Merged Result tree with two branches:

- Module mount table. Module mount table node lists the in the project used modules. See the sample of changes in the module mount table.
- Project options. Project options node shows if any project options is changed or not.

Sample of changes in the module mount table

Three changes occurs in this sample: Profile A module is updated to a newer version (change c1), which has type A removed (change c2), which implies type unsetting for the typed element P2 (change c3) (see Figure 185 on page 428).

Change c1 and c2 are dependent on each other and change c2 depends on change c3. This situation is depicted in Figure 186 on page 428.

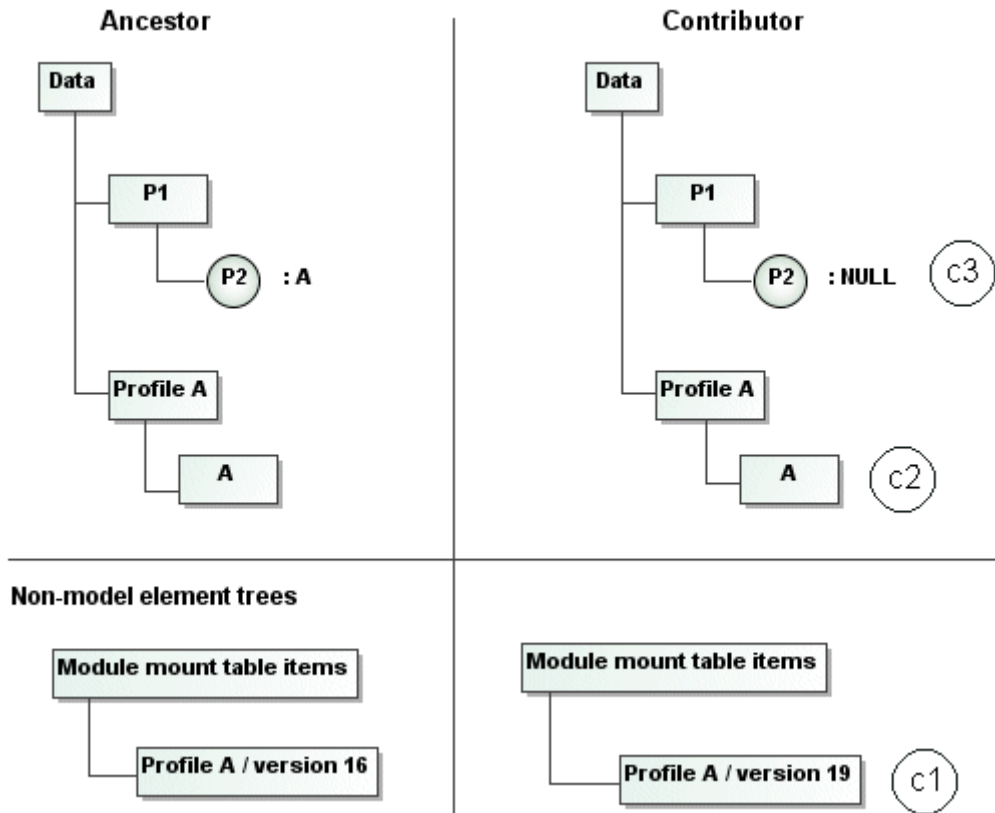


Figure 185 -- Sample of the changes in the module mount table

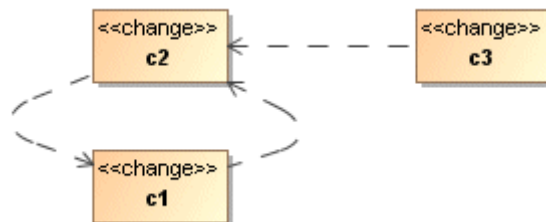


Figure 186 -- Sample of the changes in the module mount table - dependencies

Analyzing Merging Results

Merging results are presented in the **Merge** window (see Figure 187 on page 430).

The Merge window description

Look at each part of the **Merge** window separately. The **Merge** window consists of the following parts:

- "The Merged Result tree"
- "Toolbar for displaying and navigating through changes"
- "Toolbar for accepting and rejecting changes"
- "Change legend and summary"
- "Buttons for quick navigation through conflicting changes"
- "Element Specification panel"
- "Change details panel"

Reorganizing panels of the Merge window

Panels of the **Merge** window can be organized to your desired positions. The following changes can be done to the **Merge** window:

The **Summary/Legend** panel can be turned on or off using the **Summary/Legend** button, which is on the left side of the **Summary/Legend** panel (see Figure 188 on page 431).

The **Change details** and **Specification** panels can be maximized, toggled to floating or auto-hide windows using buttons, located at the top-right side of the panel (see Figure 189 on page 432).

You can also manage panels of the **Merge** window, using shortcut menu. To invoke it, right-click on the title bar of the window. See the location of the shortcut menu invocation in Figure 189 on page 432 and in Figure 191 on page 434.

To reset the position of the panels of the **Merge** window panel, from the panel shortcut menu select the **Reset Windows Configuration** command or press the **Reset Windows Configuration** button at the top of the **Merge** window (see Figure 192 on page 435).

For information about managing diagram tabs, see "Viewing changes in diagrams" on page 454.

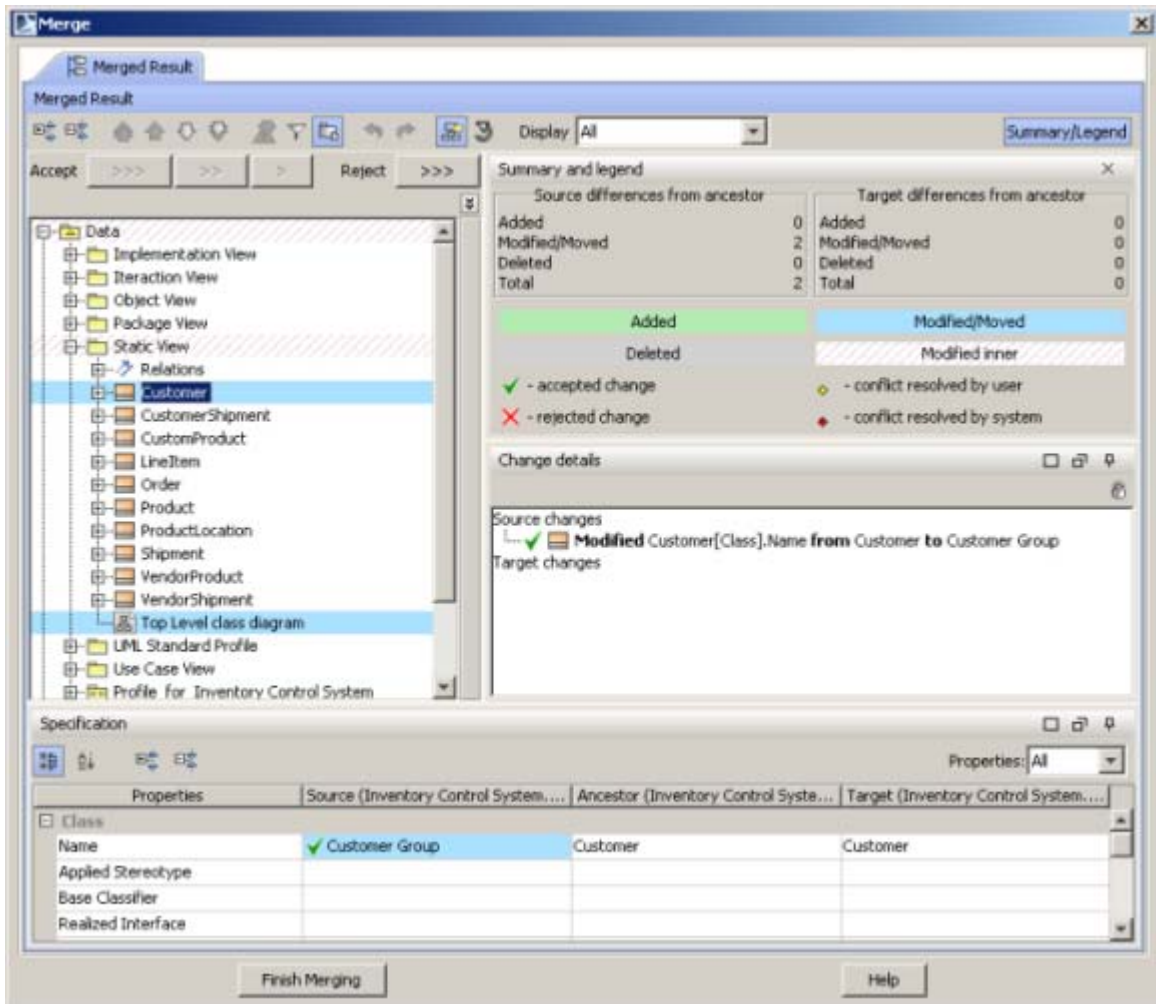


Figure 187 -- The Merge window

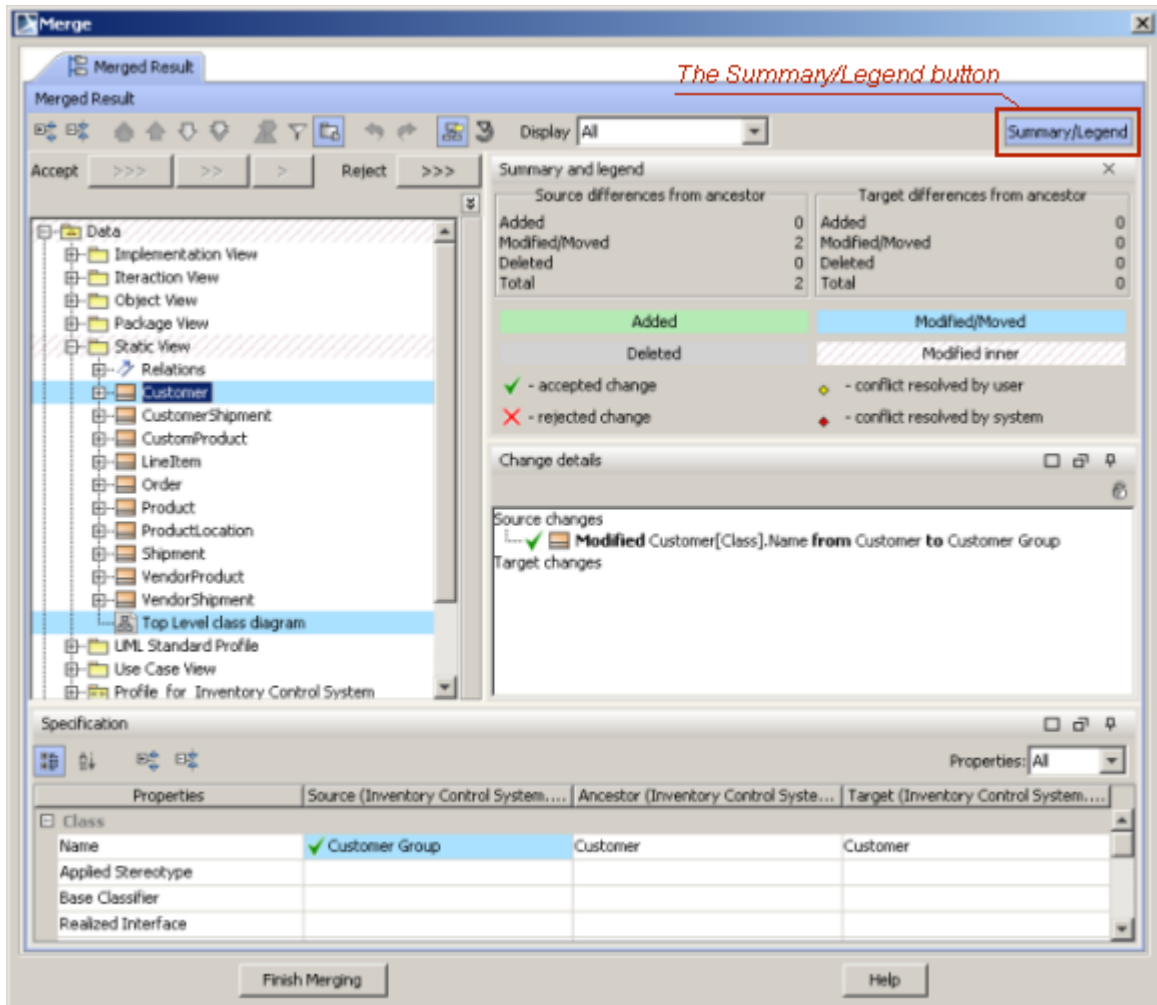


Figure 188 -- The Summary/Legend button in the Merge window

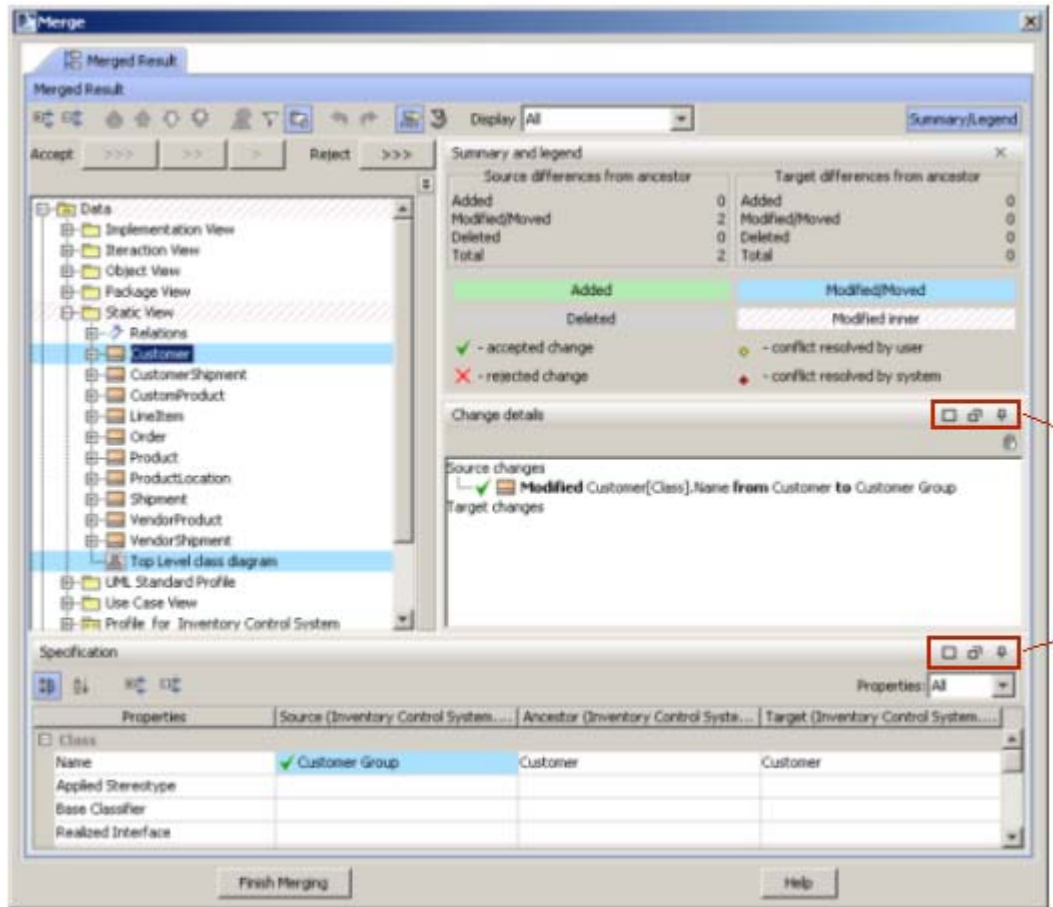


Figure 189 -- Buttons to manage the Change details and the Specification panels in the Merge window

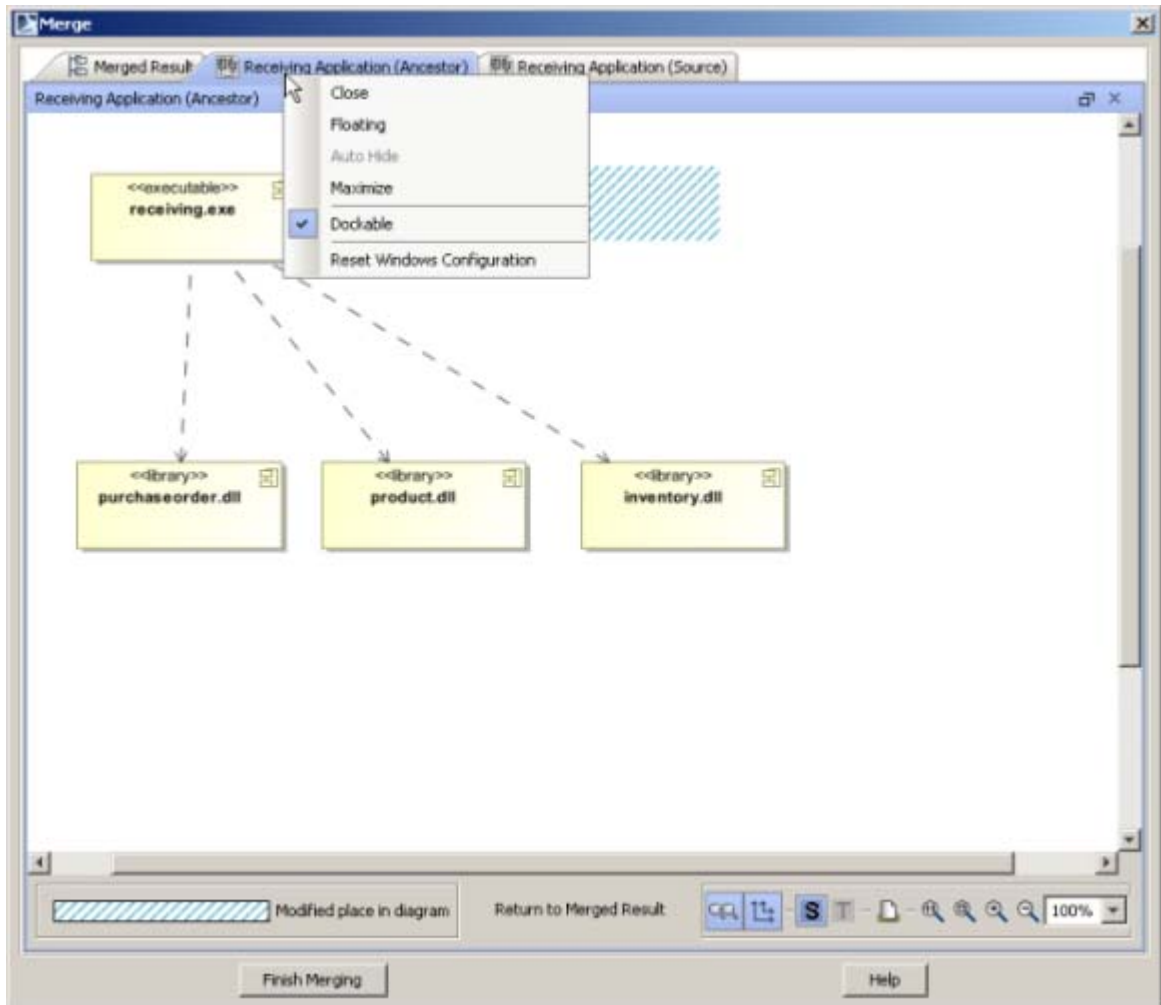


Figure 190 -- Shortcut menu of the Merge panels

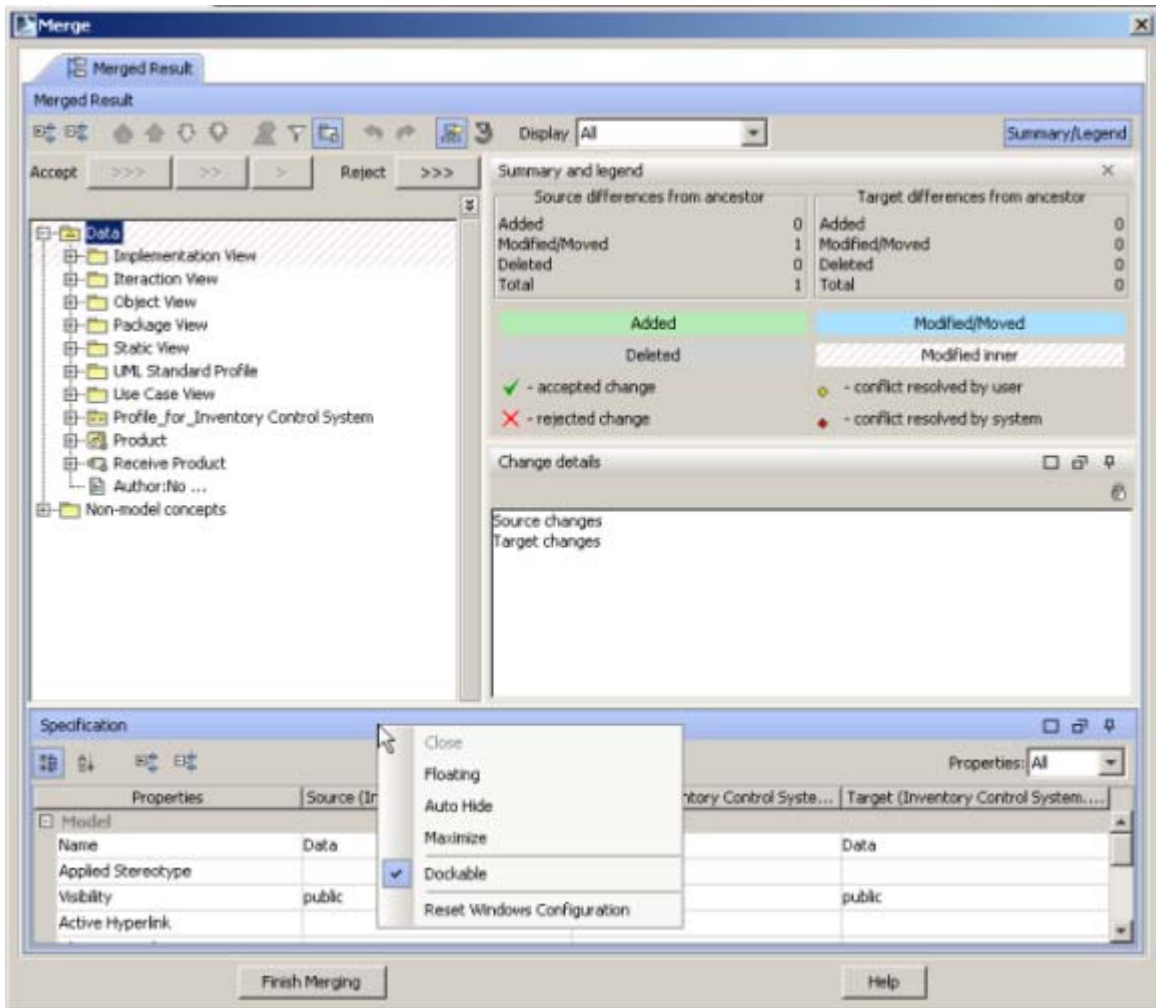


Figure 191 -- Shortcut menu of the Specification window

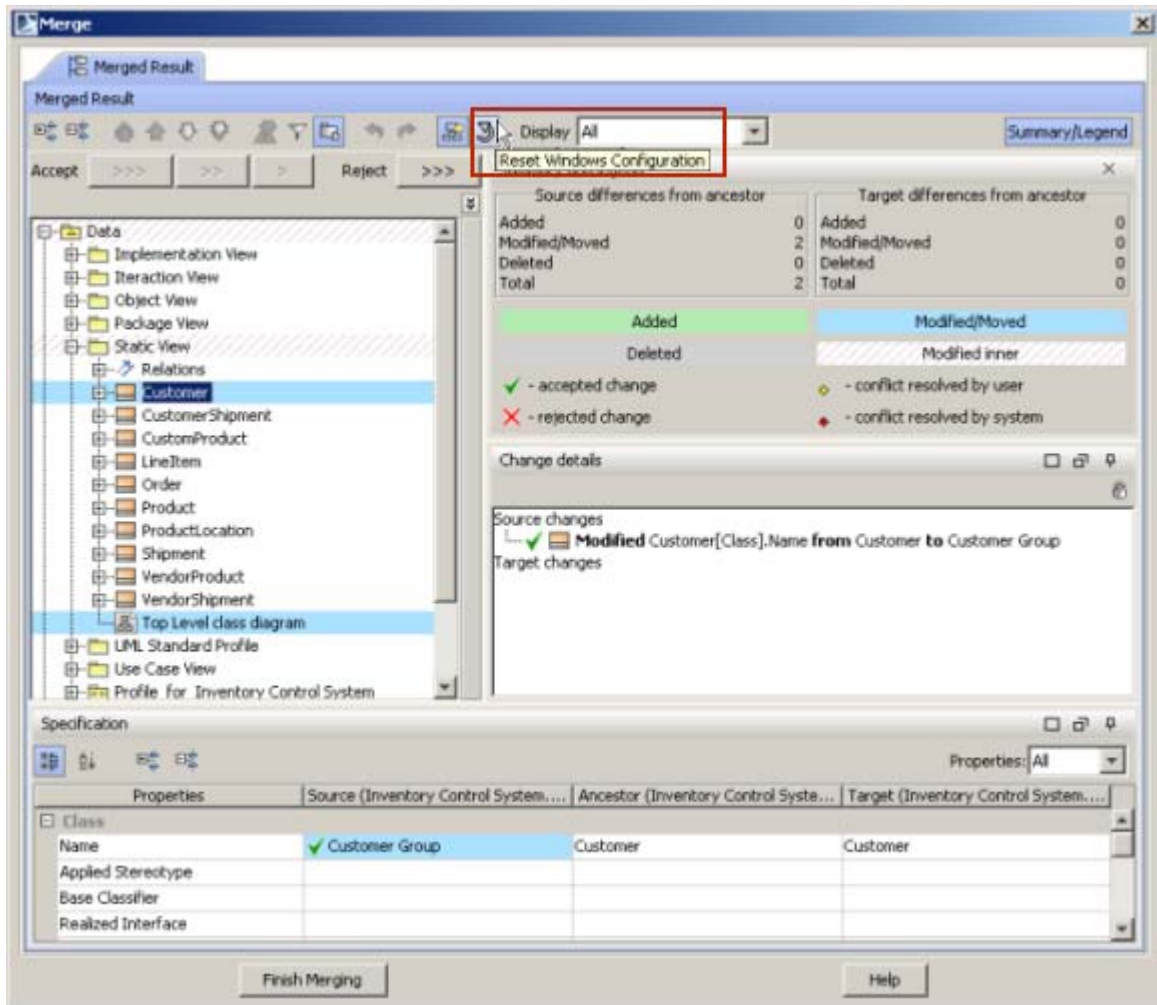


Figure 192 -- The Merge window, Reset Windows Configuration button

The Merged Result tree

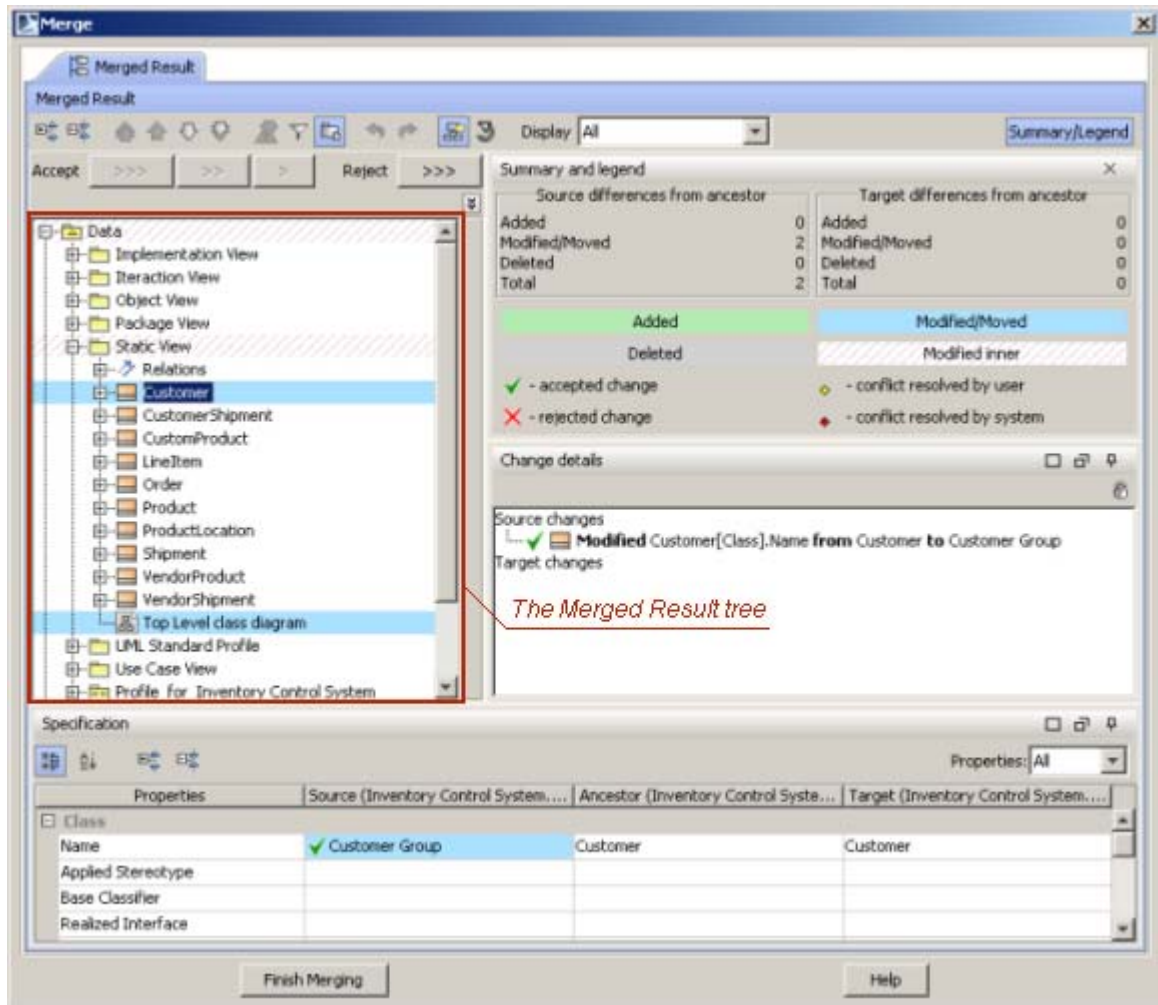


Figure 193 -- The Merged result tree



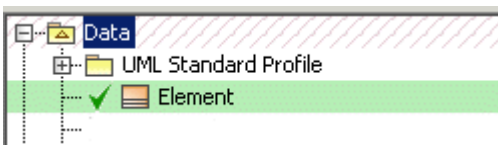


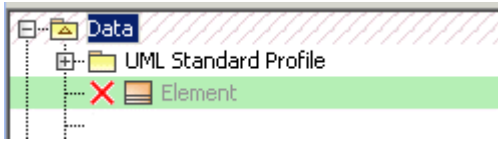
The merged result tree combines both containment and change trees. The following types of changes are displayed in the merged result tree:

1. "Addition changes" on page 438
2. "Deletion changes" on page 439
3. "Move changes" on page 439
4. "Order changes" on page 441
5. "Elements with modified properties" on page 441
6. "Elements with changed inner elements" on page 441.

Modification changes are displayed in the element properties panel other changes are displayed in the **Merged Result** tree.


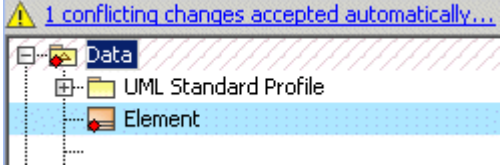

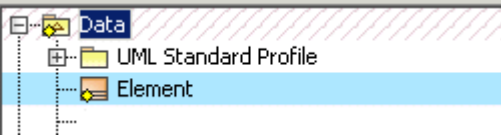
Element decoration in the Merged Result tree

A change occurs on a single element and that element has an icon indicating the state of the change (see the table below).

Decoration	Change state	Example
  Element (green tick before the element)	Accepted	 <p>New element is created in one of the contributors and its addition is accepted.</p>
  Element (red cross before element)	Rejected	 <p>New element is created in one of the contributors and its addition is rejected.</p>

For more information about accepted/rejected changes, see "Accepting or Rejecting changes" on page 425.

Changes that have conflicts are additionally decorated with icons showed in the table bellow. Parents of conflicting elements have conflict decorations too.

Decoration	Change applied by	Example
 Element (red diamond on the left-bottom corner of element)	System	 <p>Element has conflicting modification change.</p>
 Element (yellow diamond on the left-bottom corner of element)	User	 <p>Element has conflicting modification change which is resolved by the user.</p>

Addition changes

Addition changes occur when elements are created in the contributors. In the merged result tree added elements are highlighted with green color (see Figure 194 on page 438).

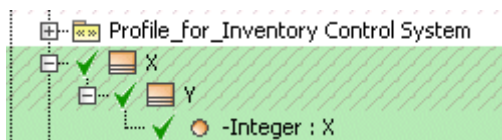


Figure 194 -- Addition change

Deletion changes

Deletion changes occur when elements are deleted in the contributors. Deletion changes are highlighted with grey color (see Figure 195 on page 439).



Figure 195 -- Deletion changes

Move changes

Move changes are highlighted using the same blue color that is used to highlight modification changes, because move change is just a kind of modification change. Additionally, move changes are displayed with arrows before the moved element icon. In the following example, element X is moved from package A to package B. The illustration shows the initial situation when where the move change is accepted and element X is owned by package B.



Figure 196 -- Move changes

Navigation from the original to the new location (and vice versa) for elements that have been moved is possible from the moved element shortcut menu (see Figure 197 on page 440 and Figure 198 on page 440).

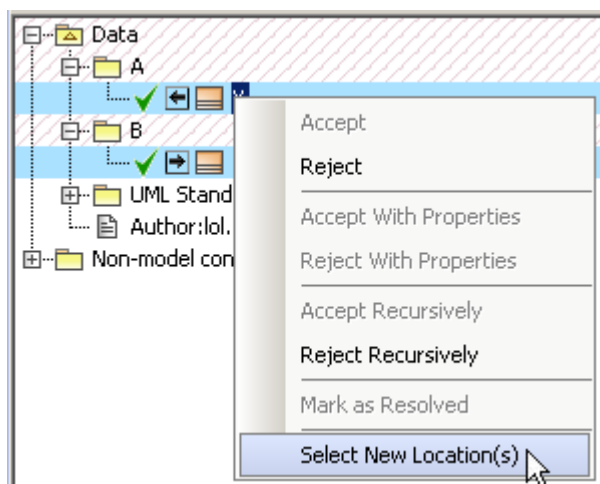


Figure 197 -- Navigation to the new element location

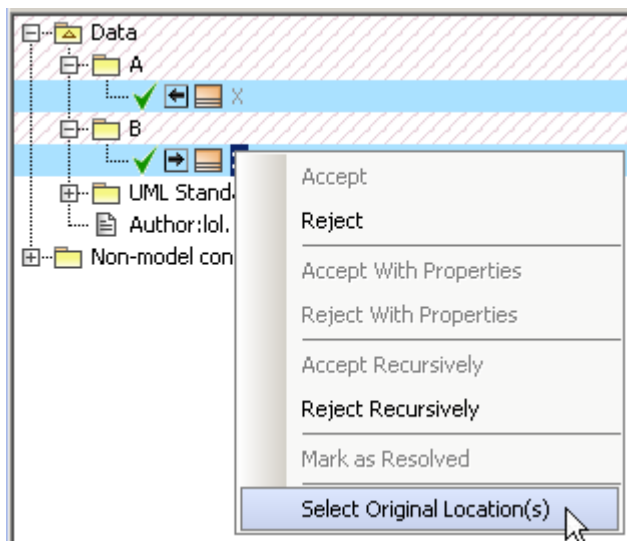


Figure 198 -- Navigation to the original element location

Order changes

Order changes occur on elements such as attributes, operations, and other ordered elements. Even if a single element in a collection has changed its place, the order change is applied to the entire collection. After accepting the change, all elements in the collection is reordered so that it would be the same as in one of the contributors the change occurred.

An element can have several ordered collections. This means several order changes can occur for a single element. Changes in the element will be highlighted using the same blue color that is used to highlight modification changes.

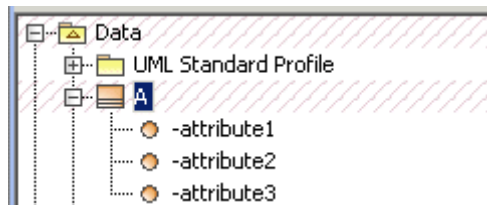


Figure 199 -- Order changes

Elements with modified properties

Modification changes occur when element properties changed in the contributors. Elements whose properties changed are highlighted in blue.

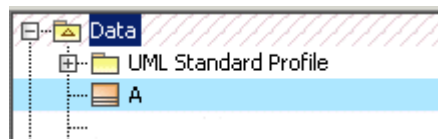


Figure 200 -- Elements with modified properties

Elements with changed inner elements

Elements whose inner elements (in any nesting level) have pending changes are highlighted using red dashes as shown below. If the element has pending property modification changes, then the modification highlight color is mixed with changed inner element highlight, i.e. the element is highlighted using blue color with red dashes.

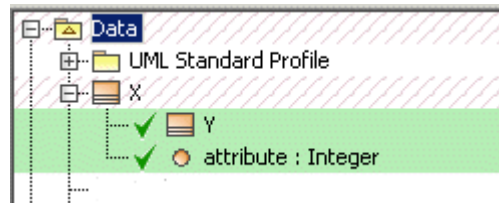


Figure 201 -- Element "X" with changed inner elements "Y" and "attribute"

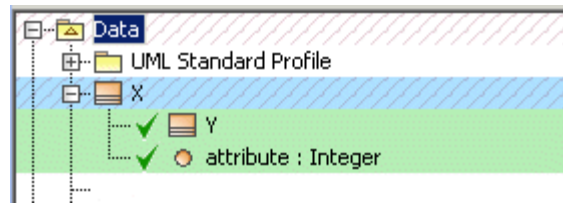


Figure 202 -- Element "X" with changed inner elements "Y" and "attribute" and modification change

Toolbar for displaying and navigating through changes

The toolbar for displaying and navigating through changes is located at the top of the **Merge** window. See the highlighted area in Figure 203 on page 443.

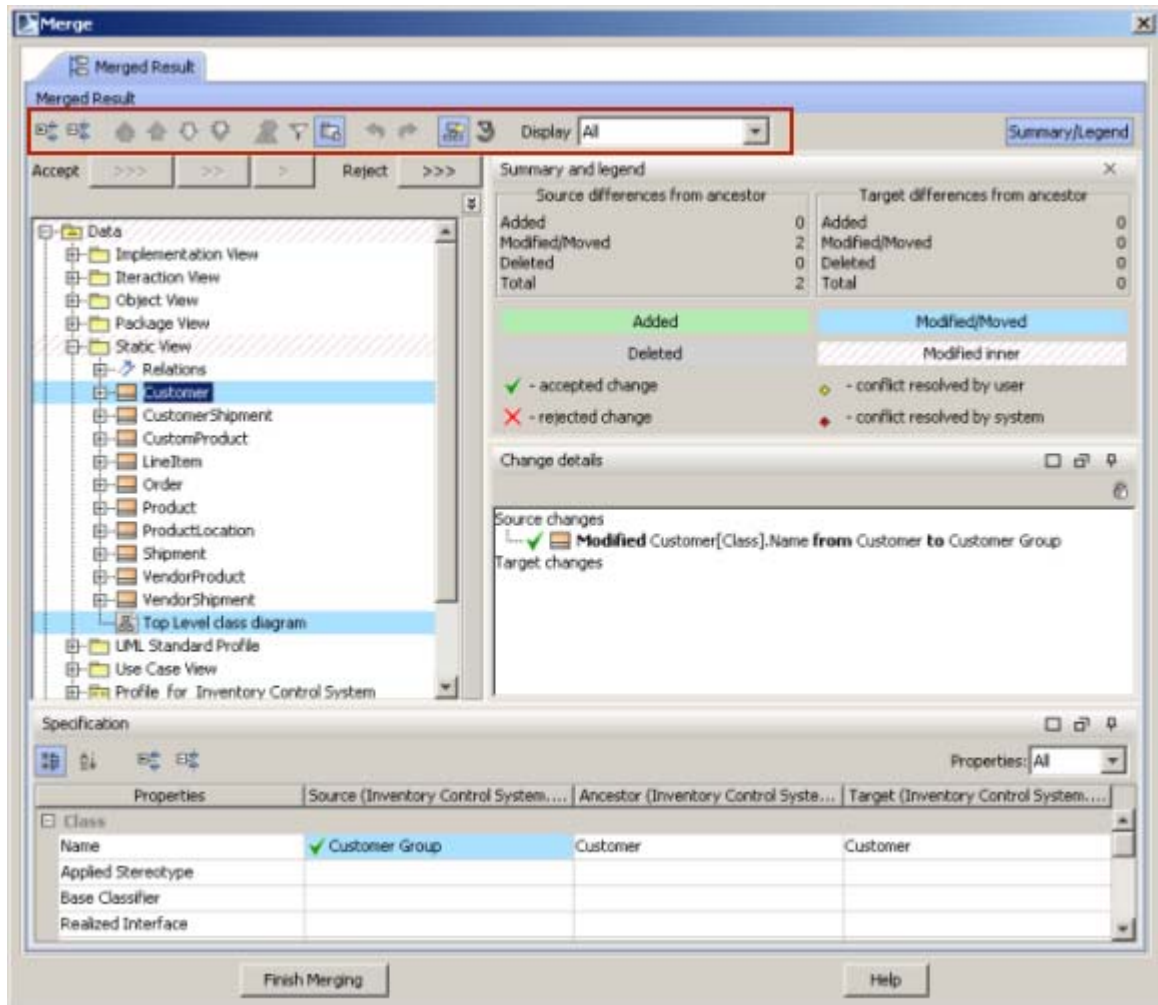




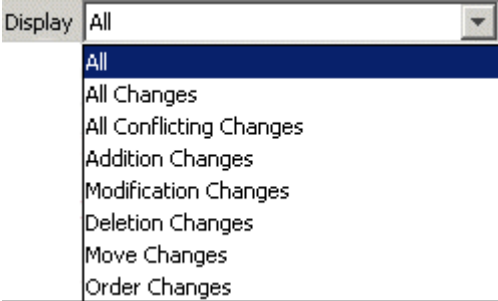


Figure 203 -- Toolbar for displaying and navigating through changes

The toolbar buttons are used to navigate through the **Merged Result** tree and the **Specification** panel:

Button	Title	Function
	Expand	Expand all nodes in the merged result tree.
	Collapse	Collapse all nodes in the merged result tree.
	Go To First Change Alt+Home	Select the first model change in the merged result tree. Note: The Go To First Change button is disabled if the first change is already selected in the merged result tree.
	Go To Previous Change Alt+up arrow	Select previous change in the merged result tree.
	Go To Next Change Alt + down arrow	Select the next change in the merged result tree.
	Go To Last Change Alt+End	Select the last difference in the merged result tree.
	Go To Next Conflict Alt+Page Down	Select the next conflict in the merged result tree. Note: The Go To Next Conflict button is disabled if there are no conflicts. For more information about navigation through conflicting changes, see “Buttons for quick navigation through conflicting changes” on page 449.
	Show auxiliary resources	Press the button to show or hide the profiles, modules with applied <<auxiliaryResources>> stereotype in the Merged Result tree (for example, UML Standard Profile).
	Undo	Undo the last action.

Button	Title	Function
	Redo	Cancel undo command.
	Filter	Press the Filter button to invoke the Items Filter dialog box. Clear the checkbox next to the it to hide element in the merged result tree.
	Annotate Merged Diagram	Press this button to annotate diagram for the post-merge review. For more information, see “Annotations in the merged diagram” on page 462.
	Reset Windows Configuration	Press the Reset Windows button to organize panels of the Merge window to its original location. For more information about resetting windows configuration, see “Reorganizing panels of the Merge window” on page 429.
		<p>List filter options for types of change in the merged result tree:</p> <ul style="list-style-type: none"> • All. This is the default value. Display all elements in the Merged Result tree. • All Changes. Display elements with changes or have inner changes. • All Conflicting Changes. Display elements that have conflicting changes or inner conflicting changes. • Addition Changes. Display elements that are added to the model or have inner addition changes. • Modification Changes. Display elements that have been modified or have inner modification changes. • Deletion Changes. Display elements that have been deleted or have inner deletion changes. • Move Changes. Display elements that have been moved or have inner move changes. • Order Changes. Display elements whose order has been changed or have inner order changes.

Toolbar for accepting and rejecting changes

The toolbar located above the merged result tree is used for accepting and rejecting changes.

Accept and reject buttons are described in the table below and more about accepting and reject you can see “Accepting or Rejecting changes” on page 425.

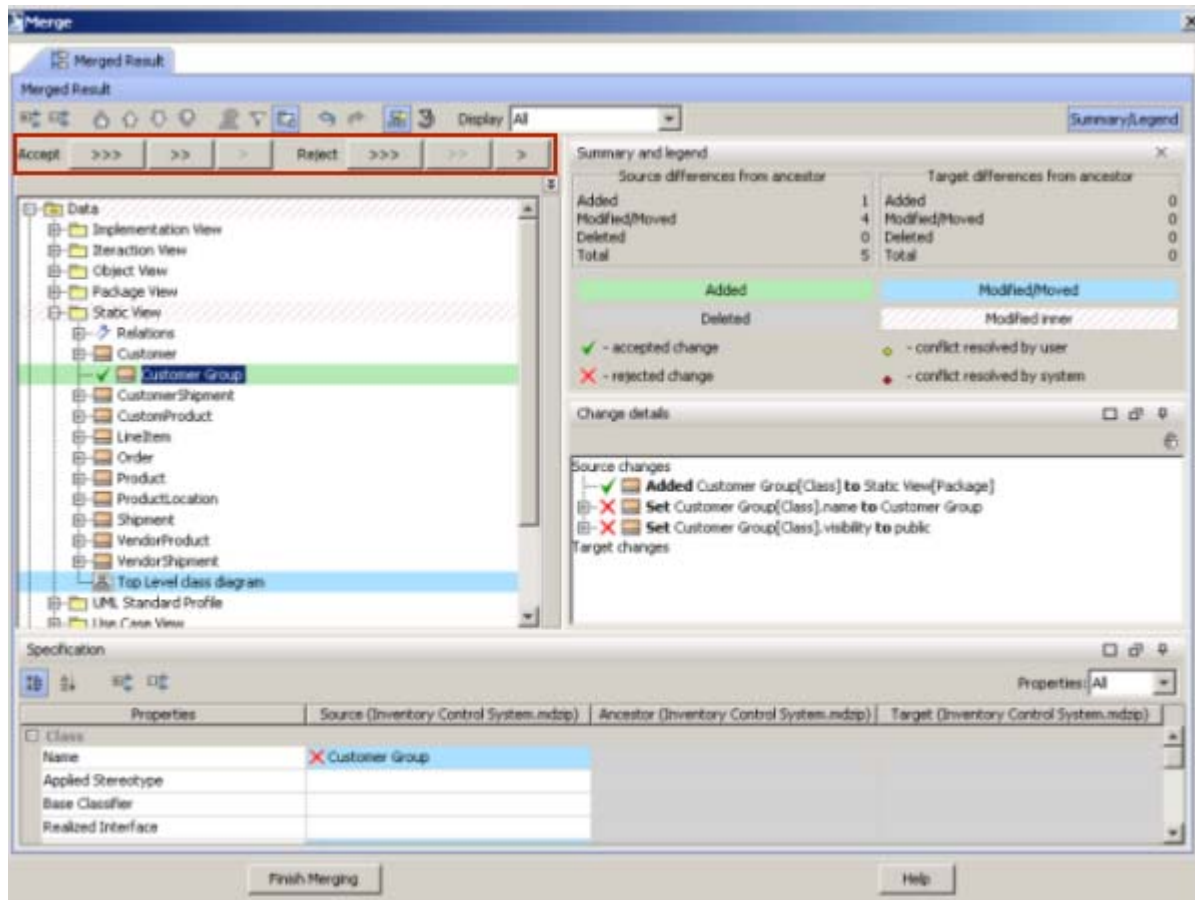




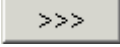
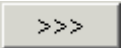
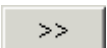



Figure 204 -- The toolbar for accepting and rejecting changes

	Button	Title	Function
<i>Buttons for accepting changes</i>		Accept the selected change, its property changes and all subelement changes	Accept all changes from the left and right contributors starting from the selected element. E.g. if the <i>Data</i> model is selected, then all changes for the whole project will be accepted.
		Accept the selected change its property change	Perform the same function as the  button, except that only the selected change and its metaproperty changes are accepted.
		Accept the selected change	Perform the same function as the  button, except that only the selected change is accepted.
<i>Buttons for rejecting changes</i>		Reject the selected change, its property changes and all subelement changes	Reject all changes from left and right contributor starting from the selected package. E.g. if the <i>Data</i> model is selected, then all changes for the whole project will be rejected.
		Reject the selected change its property change	Reject the selected change and its metaproperty changes.
		Reject the selected change	Reject the selected element change.

Change legend and summary

There is a legend on the right of the merged result for counting differences from ancestor and to display the meaning of colors that are used to mark changed elements.

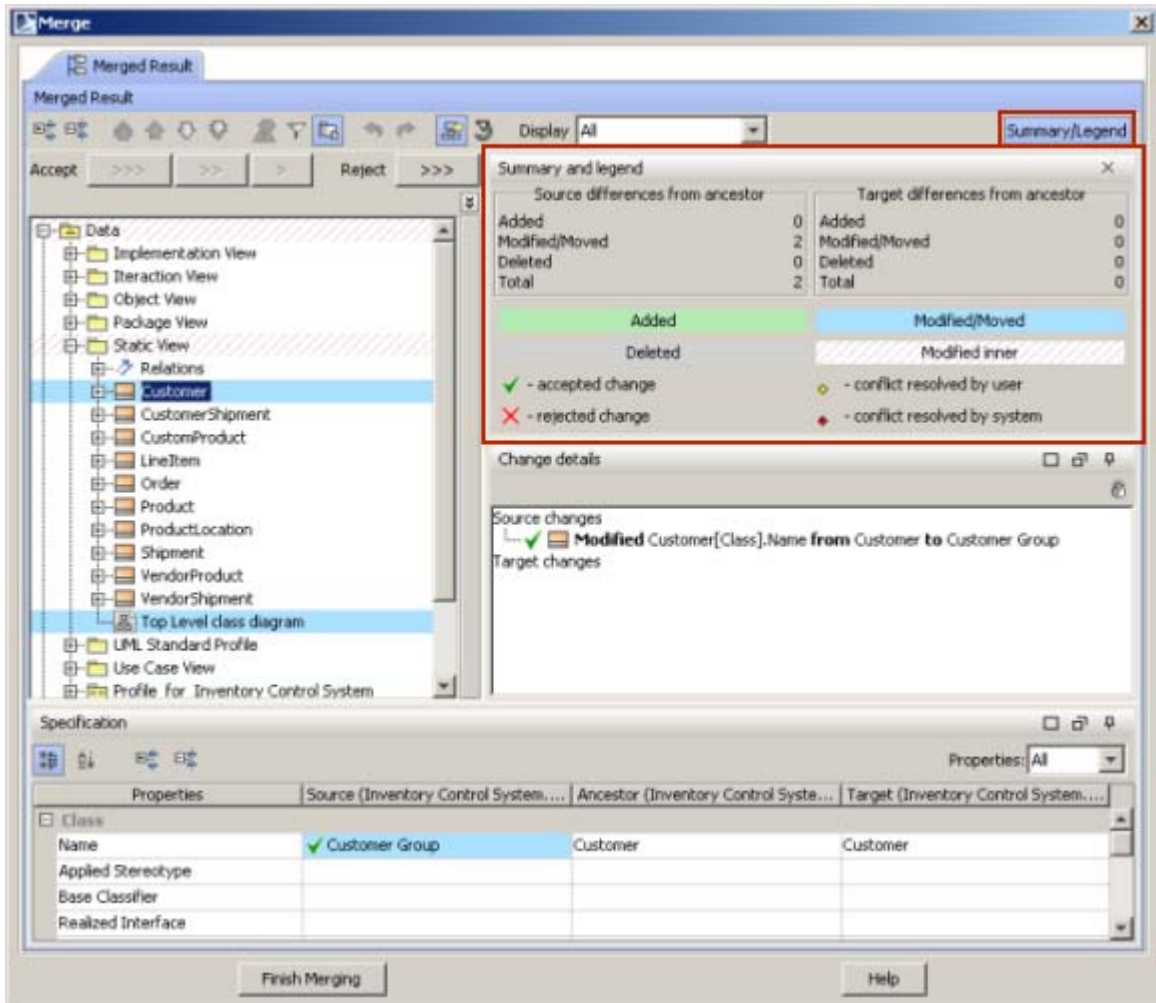


Figure 205 -- Change legend and summary

Press the **Summary/Legend** button at the top of the change legend to hide or show the summary panel.

Buttons for quick navigation through conflicting changes

Let's analyze the quick navigation through the automatically accepted conflicting changes in the **Merge** window (see Figure 206 on page 449).

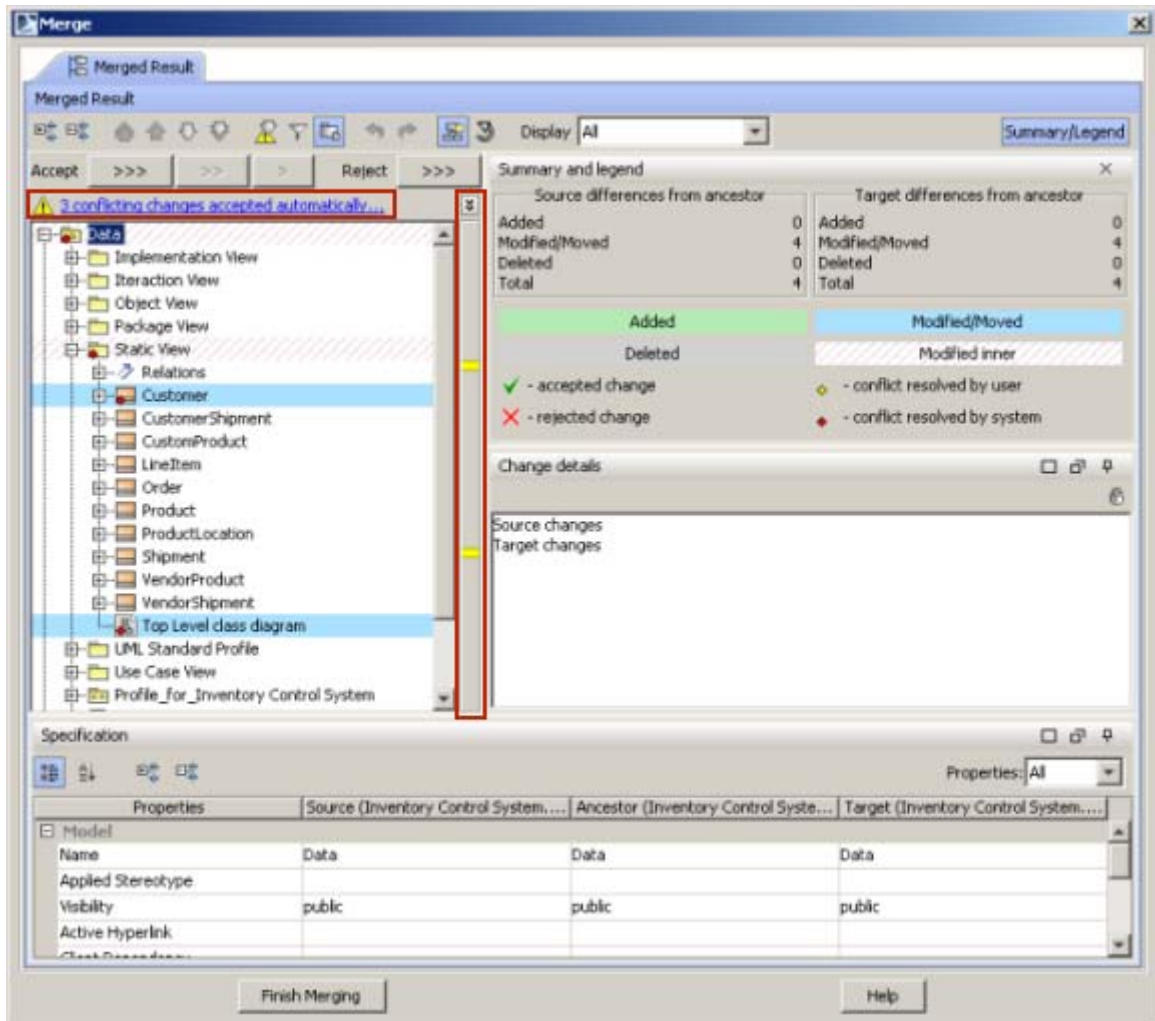


Figure 206 -- Buttons for quick navigation through conflicting changes

The panel to the right of the merged result tree displays yellow buttons representing conflicting changes. Press the yellow button to select the automatically accepted conflicting changes in the merged result tree.

A message with a warning icon is displayed in the **Merge** window stating the number of conflicting changes that are accepted from the target. This message is located above the **Merged Result** tree (see Figure 206 on page 449).

Element Specification panel

Modification changes can only occur in properties. Modification and move changes (which are a certain type of modification changes) are displayed at the bottom of the panel, the **Specification** panel (Figure 207 on page 450).

In the first column of the **Specification** panel, property titles are listed. The *Source*, *Ancestor* and *Target* columns display corresponding change properties from both contributors and the ancestor.

Changed properties have blue background. For example, Figure 207 on page 450 shows a class has different lists of values in source and target. Class is renamed in the source to *B* class and class is renamed to *C* in the target. Green tick in the *Target* column shows that class name change was accepted.

Additions in properties have green background, deletions grey have background (just like in the Merge Result tree) (see Figure 208 on page 451).

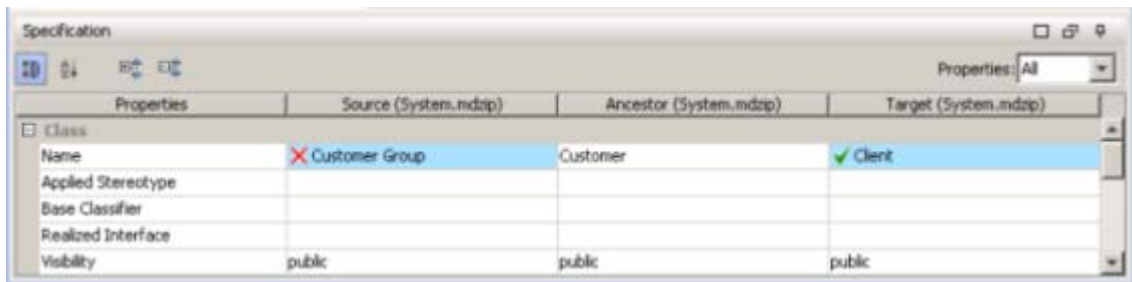


Figure 207 -- Displaying changes in element specification panel

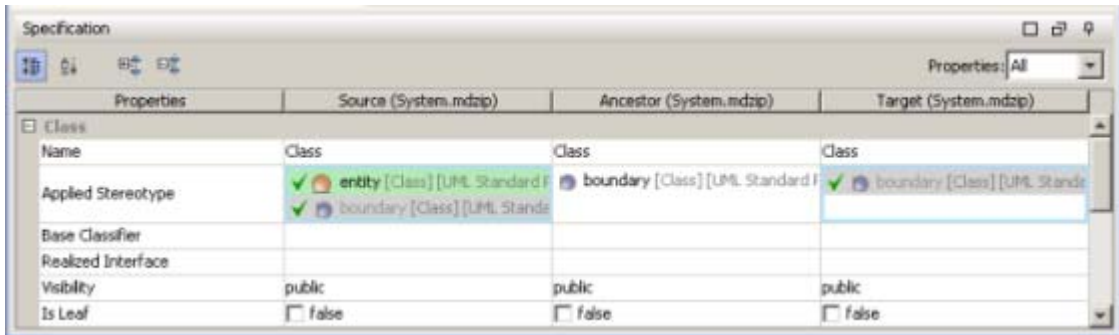


Figure 208 -- Displaying changes in the specification panel - value addition and deletion from a property

To navigate from the **Specification** panel to the **Merged Result** tree:

1. Select a property in the **Properties** column, which references other elements.
2. In the shortcut menu select the **Select in Merged Result Tree** command and then choose element to which you want to navigate (see Figure 209 on page 451).

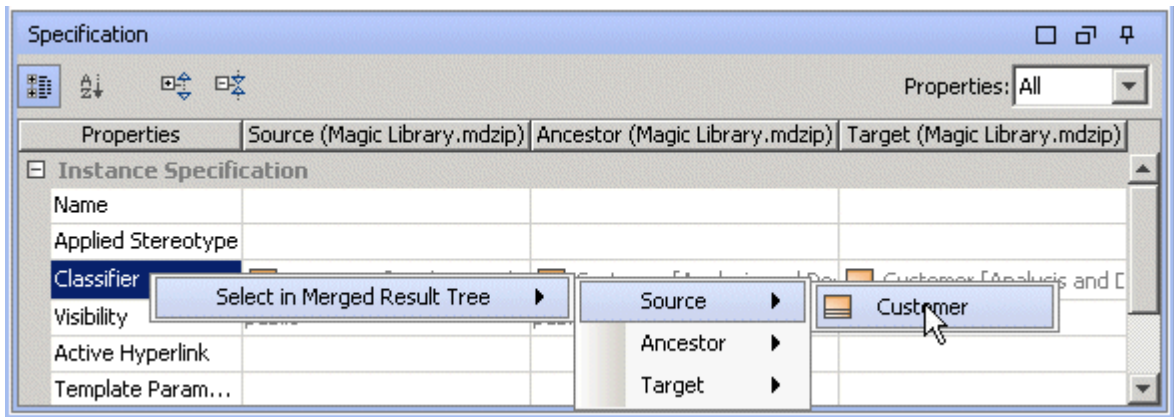


Figure 209 -- Navigating from the Specification panel to the Merged Result tree

You can also navigate to the **Merged Result** tree by invoking shortcut menu from the column cell, which references other elements. The **Select in Merged Result Tree** command has submenu items corresponding to the referenced elements for the selected cell.

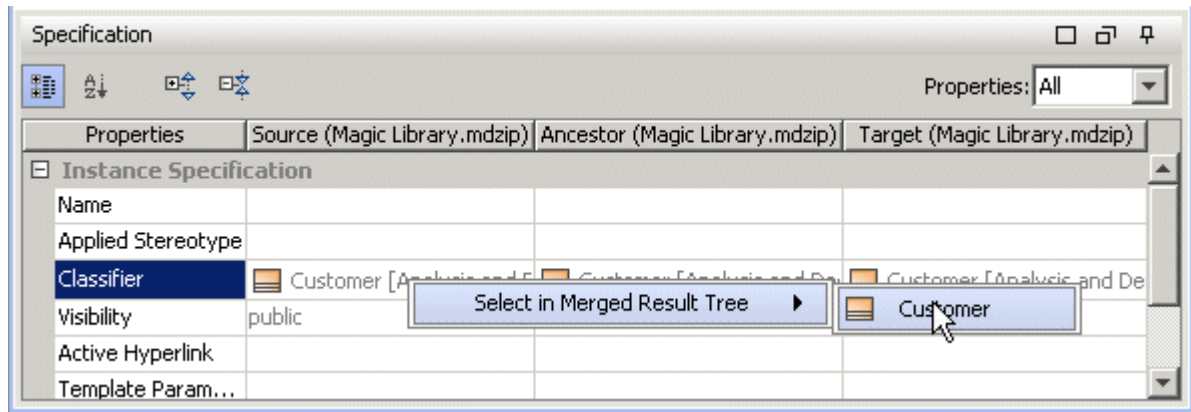


Figure 210 -- Navigating from the Specification panel to the Merged Result tree

Change details panel

The **Change details** panel is located at the bottom of the merge window. The **Change details** panel has a tree reflecting changes occurred on the element selected in the merged result tree or element properties panel. The tree has two root-level nodes:

1. Source changes
2. Target changes

The **Source changes** and **Target changes** nodes display changes occurred in the source or target respectively. If several nodes in the merged result tree are selected, then all changes occurred in the nodes are displayed in the **Change details** panel.

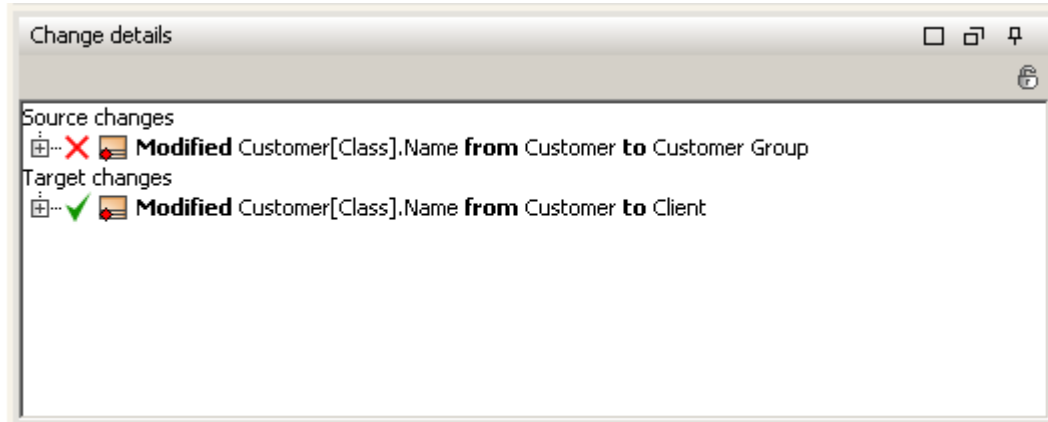


Figure 211 -- Change details panel

The **Change details** panel displays results after click on the element in the **Merged Result** tree or on property in the **Specification** panel.

Press the **Lock contents of this panel** button in the **Change details** panel to freeze the **Change details** results, that is, the last result is displayed and result is not refreshed (see Figure 212 on page 453). To unfreeze the **Change details** panel, press the **Lock contents of this panel** button again.

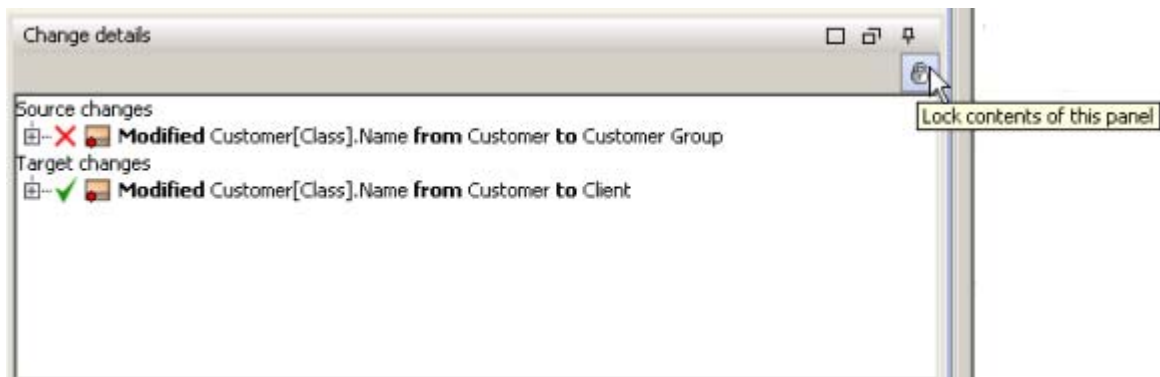


Figure 212 -- The Lock contents of this panel button in the Change details panel

Shortcut menu in the Merge window

The table bellow lists the commands of the shortcut menu, which is available in the **Merged Result** tree and **Change details** panel.

Command Name	Description
Accept	Accepts the selected change.
Reject	Rejects the selected change.
Accept With Properties	Accepts the selected change and its property changes.
Reject With Properties	Rejects the selecting change, its property changes.
Accept Recursively	Accepts the selected change, its property changes, and all subelement changes.
Reject Recursively	Rejects the selected change, its property changes, and all subelement changes.
Mark as Resolved	The Mark As Resolved command makes the change resolved by user, not the system. The conflicting change is marked as resolved by user too.
Select in Merged Result Tree / Select in Specification panel	This menu item changes depending on the type of the change.

Viewing changes in diagrams

The merge window has functionality to open merged diagrams from the source, ancestor, and target projects in separate tabs.

Opening the diagrams with changes from the **Merge** window

To open diagrams with changes double click on modified diagram in the **Merged Result** tree. Two or free views of the same diagram are opened. Ancestor view of diagram is opened always and source diagram is opened if changes were done in diagram in source project, accordingly the targed diagram is opened if changes were done in diagram in target project. See the Merge window with the opened diagrams (see Figure 213 on page 455).

You can switch between **Merged Result** window and diagram windows, by pressing tabs at the top of the **Merge** window.

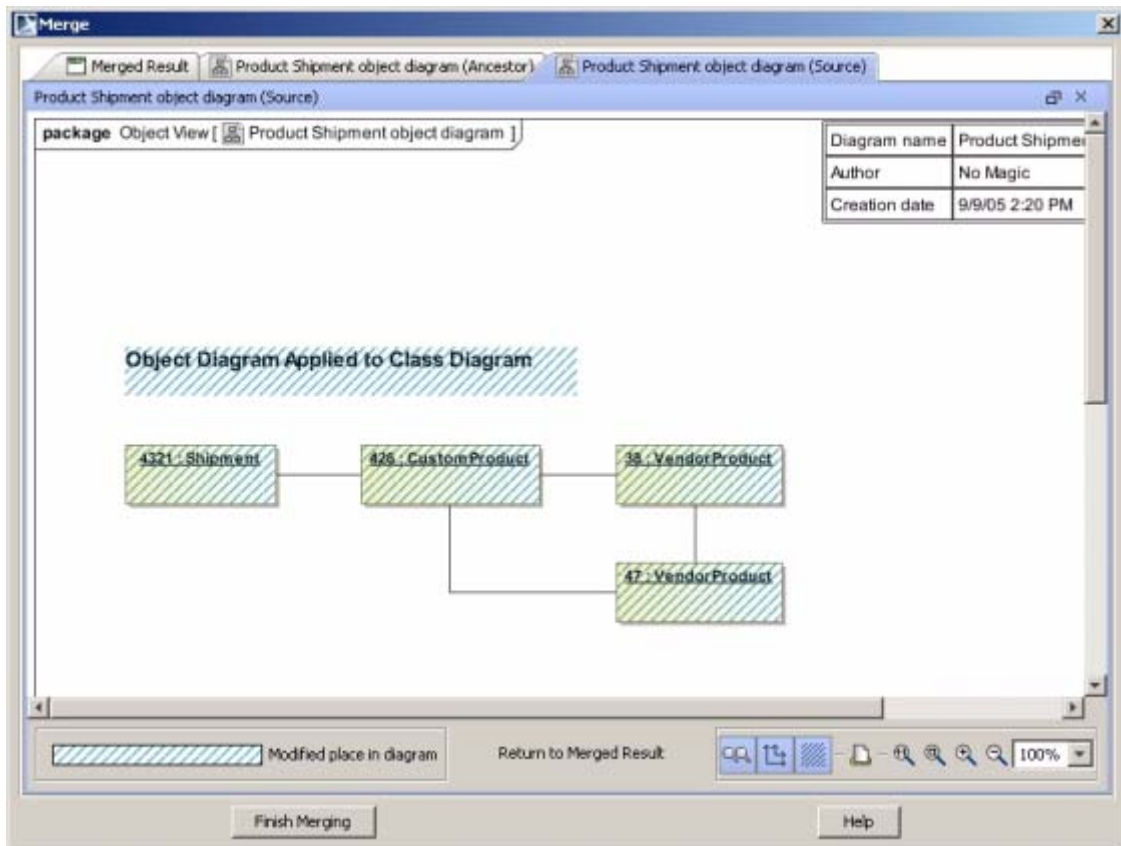


Figure 213 -- The Merge window with opened diagrams

Analyzing differences in diagrams

In opened diagrams changed diagram areas are highlighted with dashed blue background (see Figure 214 on page 456).

The following changes are highlighted in diagrams:

- The symbol or few symbols were moved or resized in diagram
- Then new symbol diagram was added

- The symbol was deleted from diagram

In summary all visual changes in diagram are highlighted.

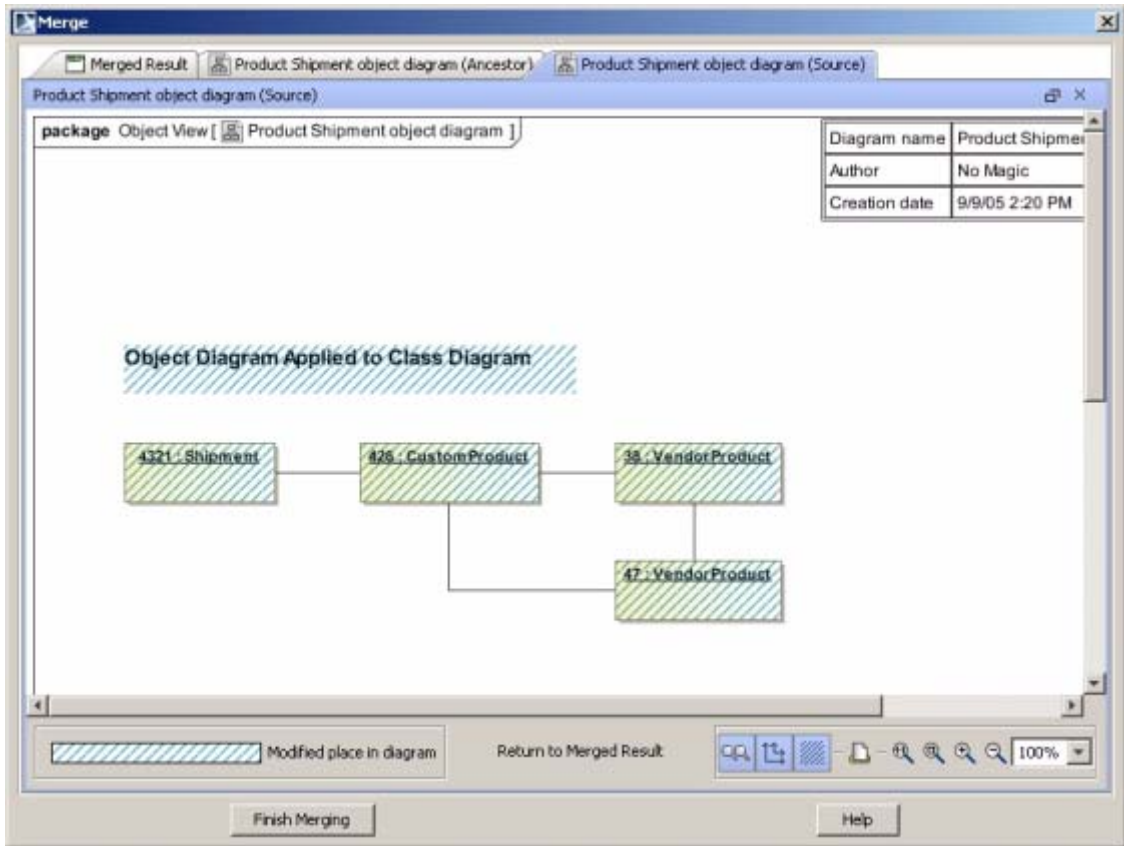







Figure 214 -- Diagram with highlighted differences

Managing the diagram view

Using the buttons located at the bottom of the diagram viewer tab (see Figure 215 on page 459) you can turn on or off difference showing, print a diagram, zoom, synchronize zooming and scrolling in diagram view. See the buttons description in the table below.

Button image and title	Description
Synchronize Zooming 	<p>When button is pressed (default value) source, ancestor, and target diagrams zooming is dependent, i.e. the same zooming operations are performed on diagrams simultaneously.</p> <p>To turn off synchronization of zooming, depress the Synchronize Zooming.</p> <p>NOTE Synchronization is valid for the source, ancestor, and target views of the same diagram.</p>
Synchronize Scrolling 	<p>When button is pressed (default value) source, ancestor, and target diagrams scrolling is dependent, i.e. the same scrolling operations are performed on diagrams simultaneously.</p> <p>To turn off synchronization of scrolling, depress the Synchronize Zooming.</p> <p>NOTE Synchronization is valid for the source, ancestor, and target views of the same diagram.</p>

Button image and title	Description
Mark Changes 	<p>Difference showing od hiding is controlled by the Mark Changes button.</p> <p>It is possible to press or depress this button in every diagram independently:</p> <ul style="list-style-type: none"> • If this button is pressed in the source diagram, then differences are shown in the source and ancestor diagrams. • If this button is pressed in the target diagram, then differences are shown in the target and ancestor diagrams. • If this button is pressed in the ancestor diagram, then differences are shown in the source, target, and ancestor diagrams. If buttons in the source and/or target diagrams were depressed previously, they are pressed after pressing this button in the ancestor diagram. <p>If difference showing is turned on for both source and target diagrams or it is turned on for the ancestor diagram, then area that highlights differences in the ancestor is combined from the areas that would be shown in the ancestor if only showing differences for source or target would be enabled.</p>
Print Diagram 	<p>To print a diagram, press the Print Diagram button. Diagrams are printed with highlighted area with the Mark Changes button is pressed (default value).</p>
Zooming 	<p>Resize the view of diagram by the zooming buttons. To synchronize or desynchronize zooming use the Synchronize Zooming button (see description above).</p>

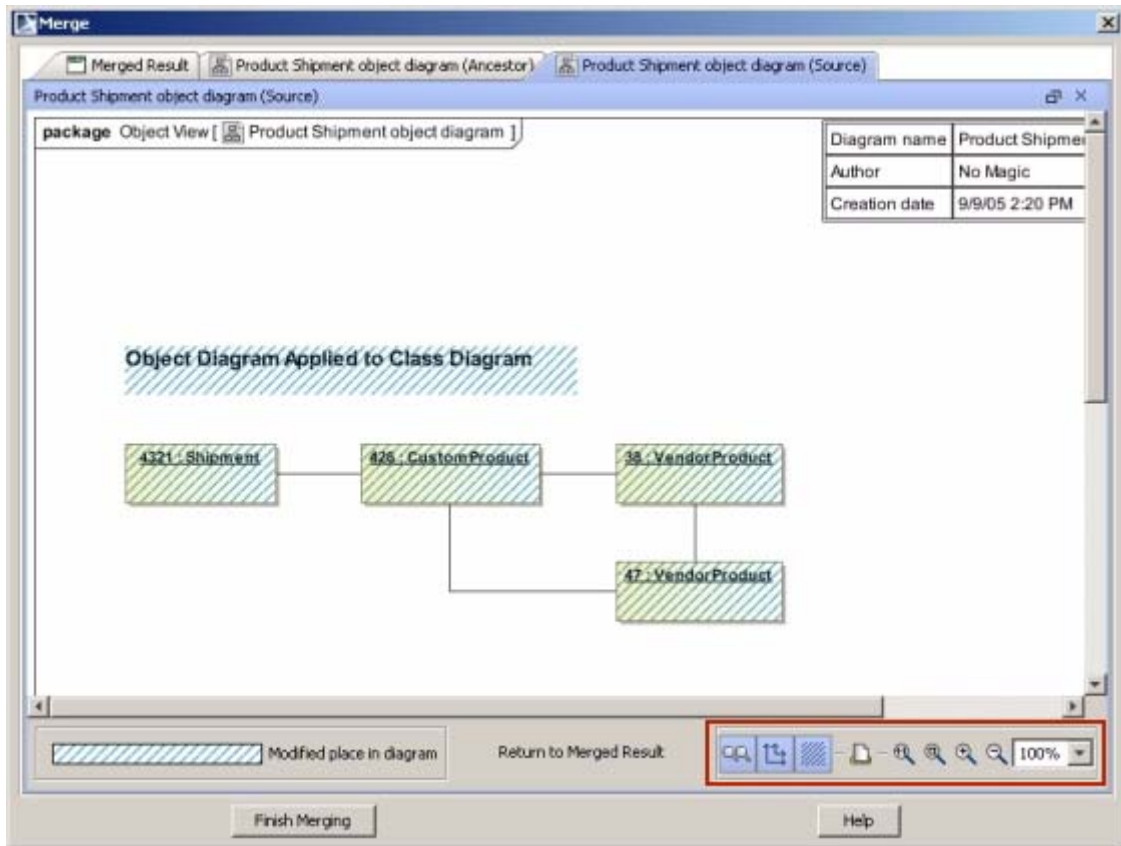


Figure 215 -- Buttons group in the diagram difference viewer

Symbol properties changes marking in the diagram difference viewer

Symbols, which are marked as changed in diagram difference viewer, but has no visual differences, may have symbol properties changes. It means that symbol properties there changes, but no affect on symbol was made.

See an example in Figure 216 on page 460 and in Figure 217 on page 461. In this example the *Shipment* class is displayed as changed in diagram, but it has no visual differences nor in Target neither in Source diagrams. The following symbol properties change was made to the *Shipment* class: in the class **Properties** dialog box, the **Show Stereotypes** option value was changed. Symbol properties

changes were made for the class and they are detected in the diagram difference viewer in order to merge them correctly.

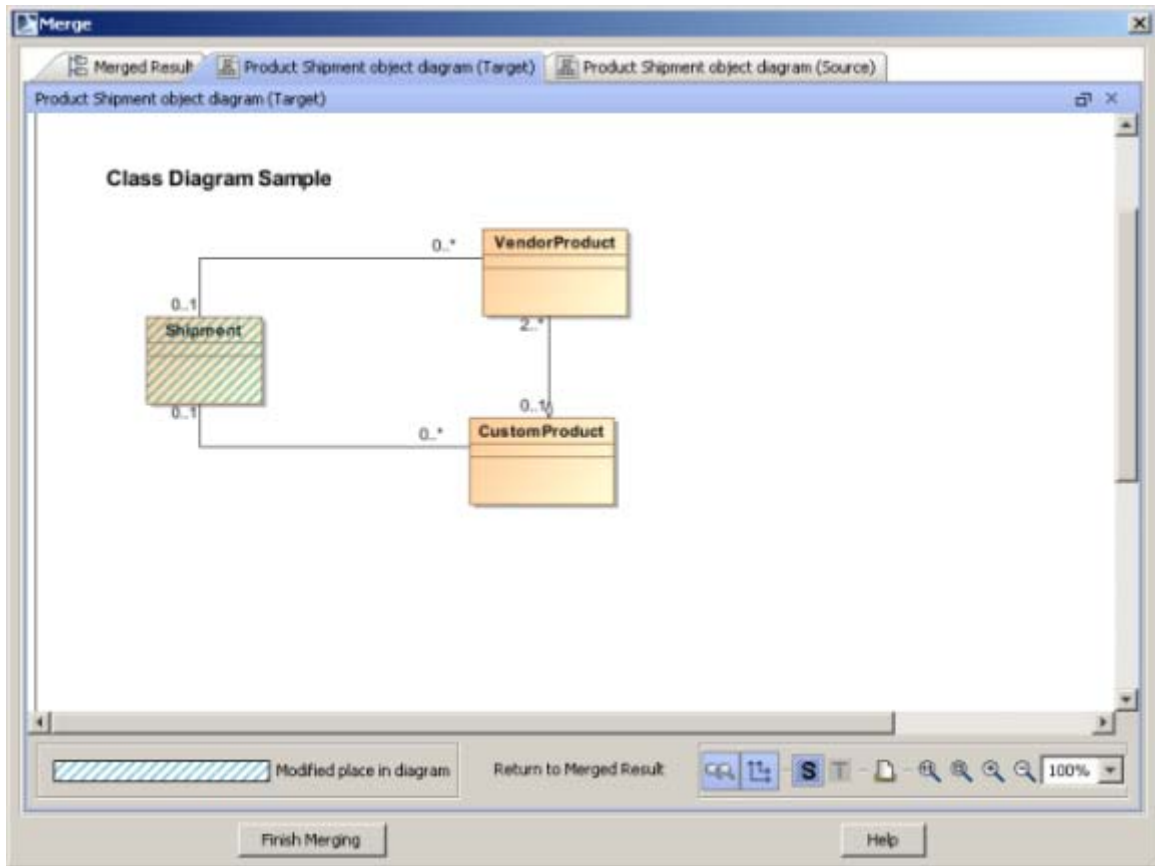


Figure 216 -- The diagram difference viewer window, displaying symbol properties changes in the Target diagram

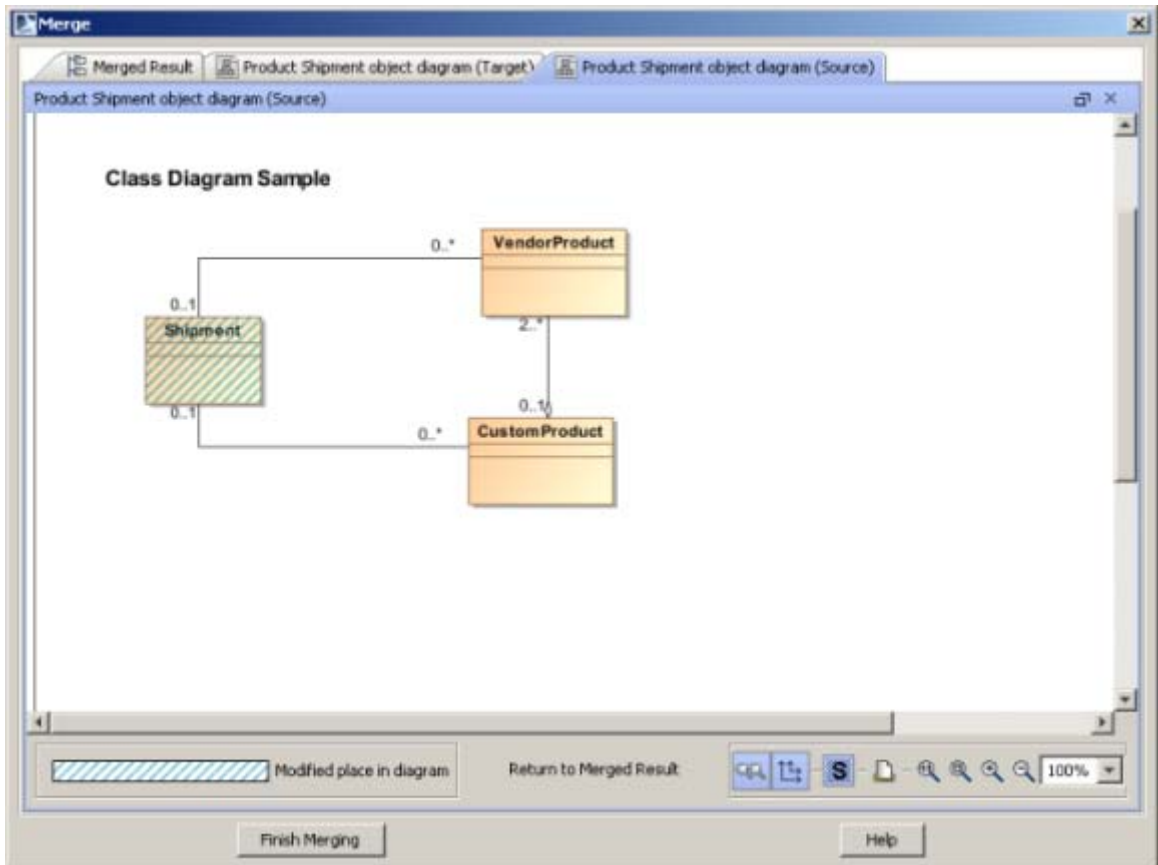


Figure 217 -- The diagram difference viewer window, displaying symbol properties changes in the Source diagram

Finishing projects merge

To finish the merge, in the **Merge** window, click the **Finish Merging** button. The question dialog box appears (see Figure 218 on page 462).

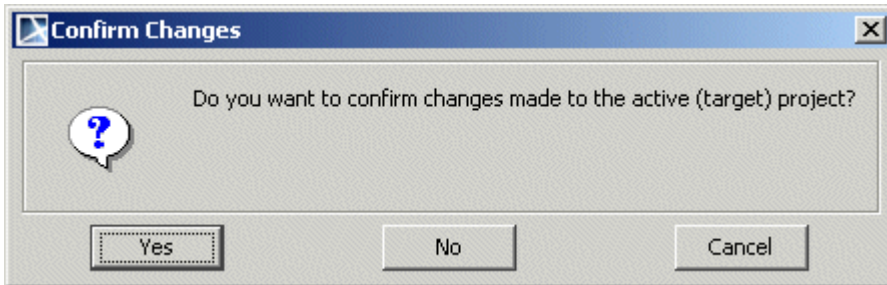


Figure 218 -- The Confirm Changes question dialog box

After pressing **Yes** in the **Confirm Changes** dialog box, project changes to the target project is confirmed.

After pressing **No**, the target project leaves not changed.


Pressing **Cancel** will cancel the **Confirm Changes** dialog box and you will be able to continue merging.

NOTE After the merge results are copied to the project, do not forget to save or to commit project to Teamwork Server.

Annotations in the merged diagram

The **Annotate merged diagram** button specifies whether merged diagrams will be annotated for the post-merge review or not. Press this button to annotate merged diagram. See the location of the button in Figure 219 on page 463.

See an example of the annotated diagram in Figure 220 on page 464.

Click the  warning at the right-bottom corner of the MagicDraw window to invoke the **Active Validation Results** window, there information about merged diagrams is represented (see Figure 221 on page 465). For more information, see "Validation" on page 601.

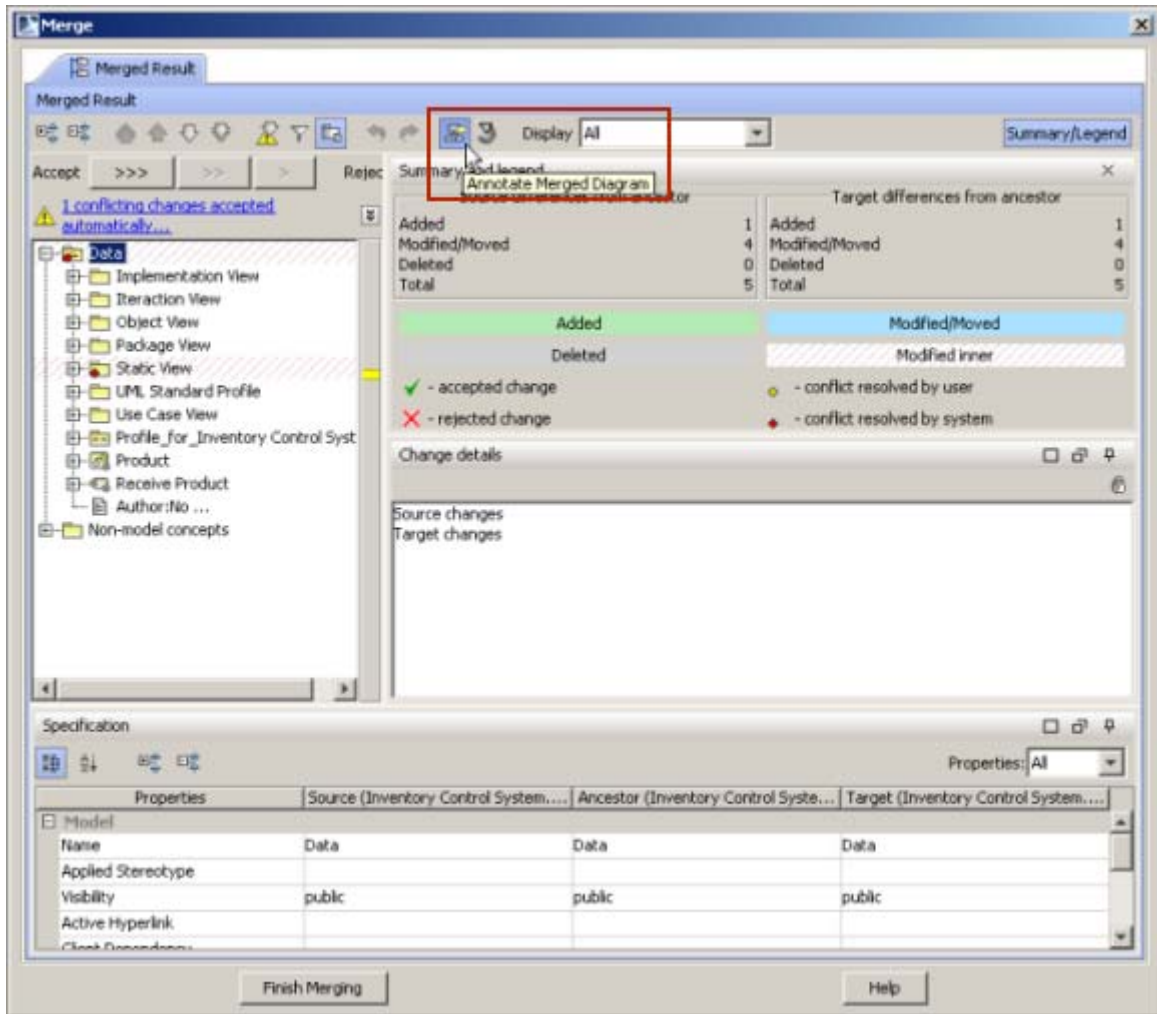


Figure 219 -- The Merge window, the Annotate Merged Diagram button

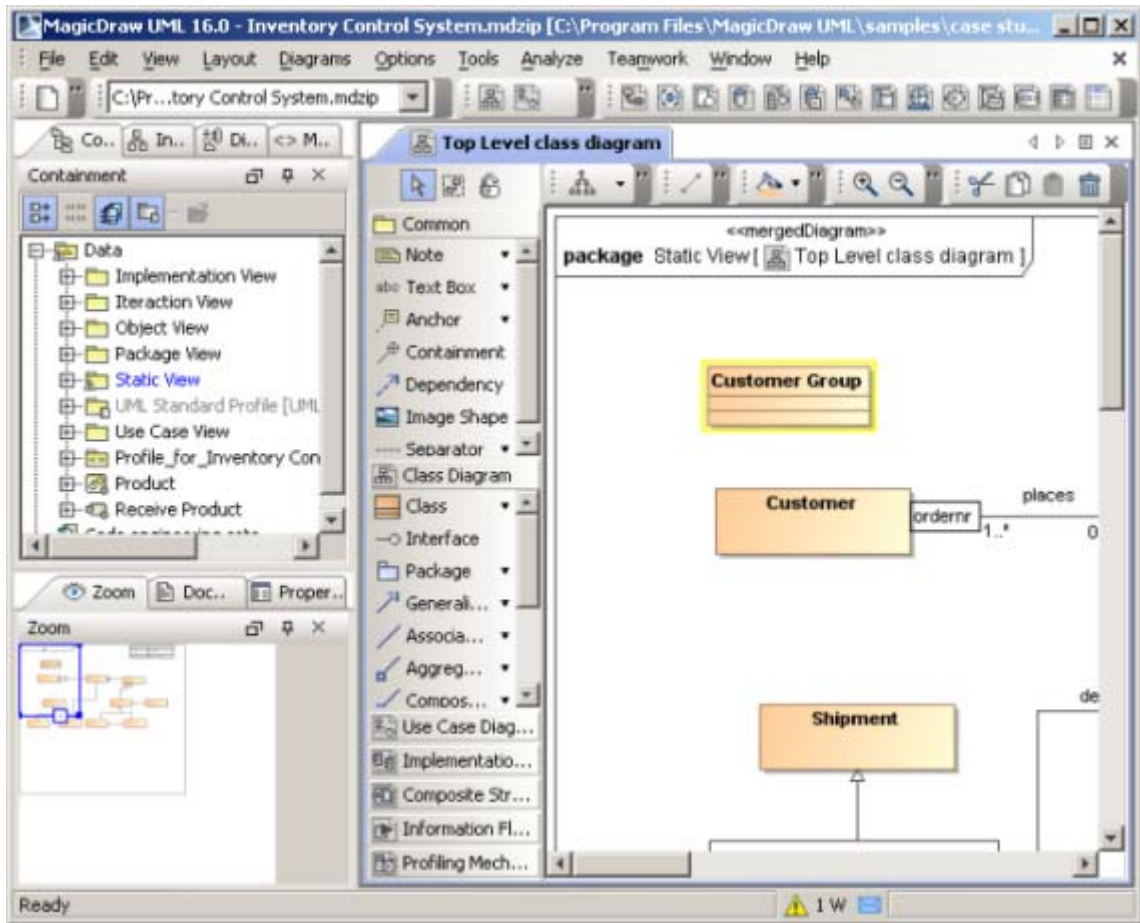


Figure 220 -- Annotations in the merged diagram

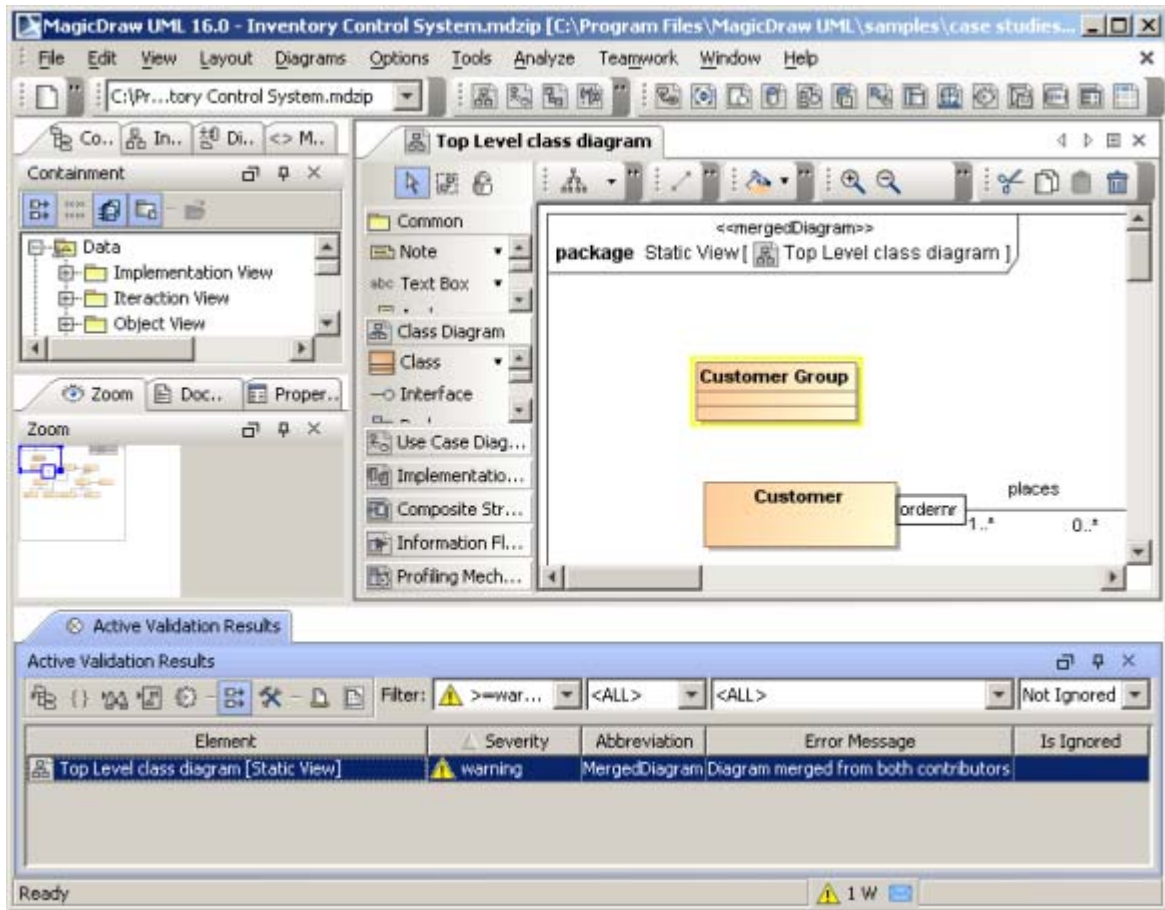


Figure 221 -- Validation results of the merged diagrams

Controlling Merge memory usage

You can customize merge memory usage by selecting the optimization option:

1. From the **Tools** main menu, select the **Project Merge** command. The **Projects Merge** dialog box opens (see Figure 222 on page 466).
2. Select the **Optimize for** option.

You can also specify the **Optimize for** option in the **Environment Options** dialog box, **General** pane, **Merge** group.

Select the **Speed** property to merge the projects faster, but it would require more memory.

If your PC doesn't have memory enough, in the **Optimize for** option, select the **Memory** property. Merge time will be slower, but memory usage will decrease.

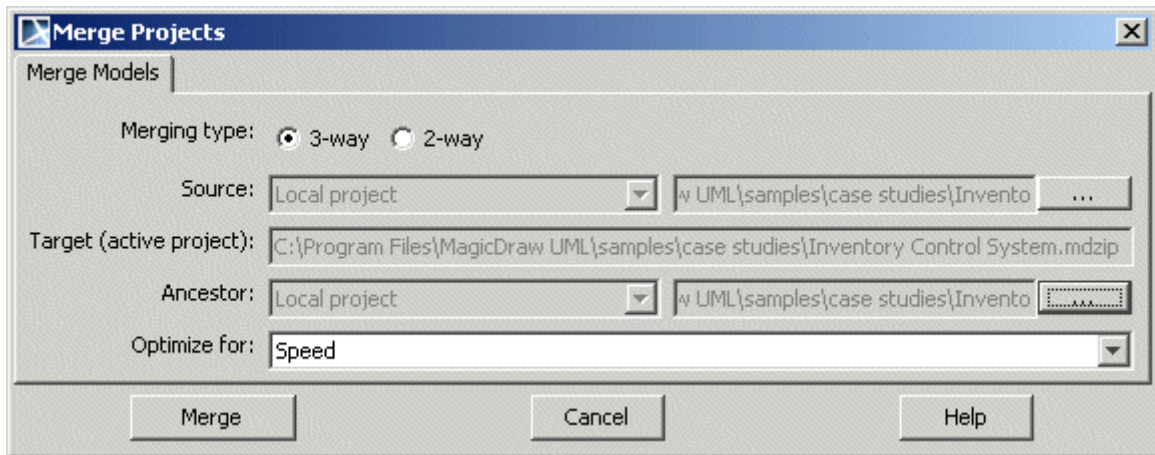


Figure 222 -- The Merge Projects dialog box

Pattern Wizard

In MagicDraw, you can find various GOF, Java, Junit, CORBA IDL, and XML Schema design patterns.

NOTE This functionality is available in Standard, Professional, and Enterprise editions only.

You can also create new patterns and edit existing ones using Java code or JPython scripts. For a detailed description, see [MagicDraw open API user's guide](#).

To open the **Pattern Wizard**

- Select **Tools** from the class shortcut menu and then select the **Apply Pattern** subcommand.

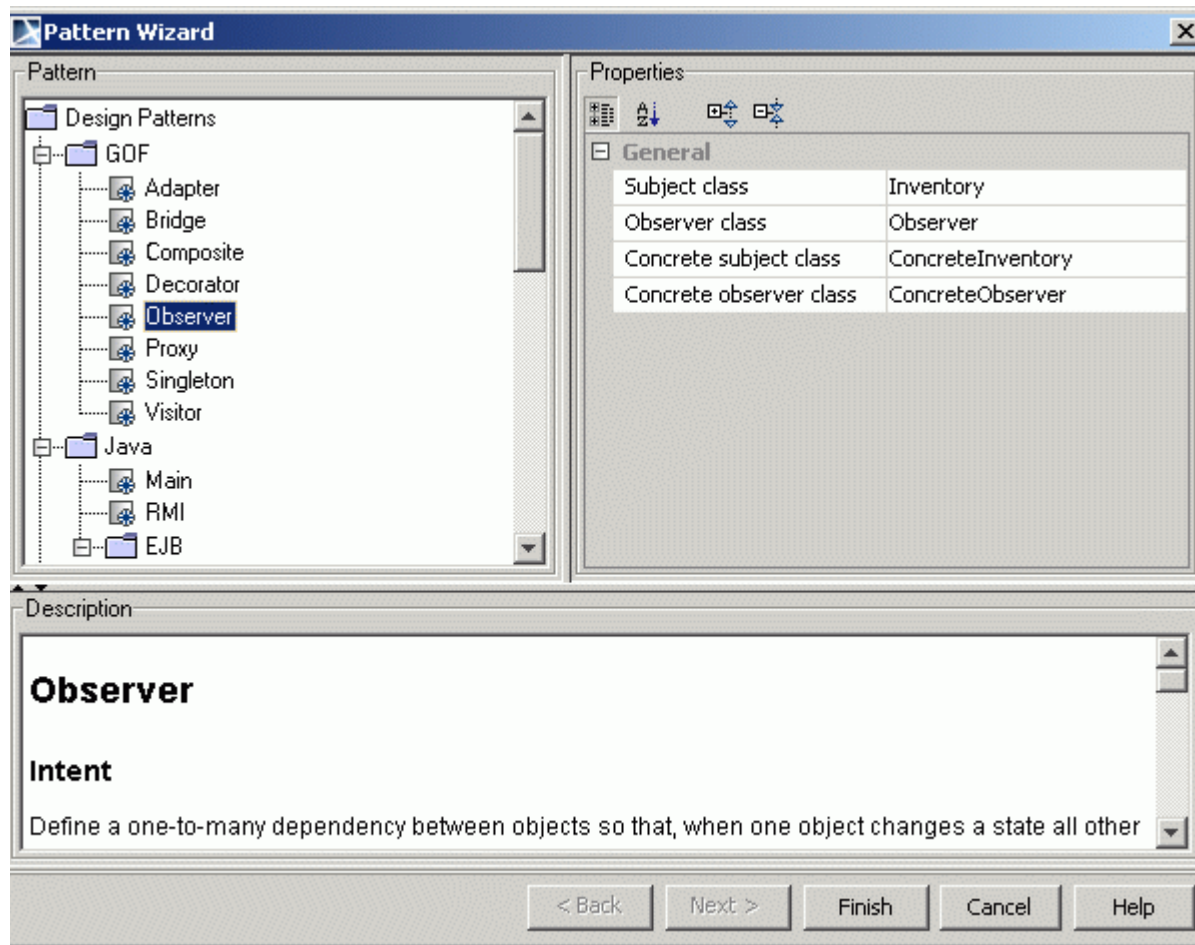


Figure 223 -- Pattern Wizard

The **Pattern Wizard** has three main collections of customizable options, which are represented by the hierarchy tree on the left side of the dialog box:

Category	Design Pattern	Properties
GOF Templates described in the Design Patterns of Reusable Object-Oriented Software	Adapter	Interface Class Adapter Class Adaptee Class NOTE The Next > button is activated. In the Adapter Operations screen, add or remove operations you want to use.
	Bridge	Abstraction Implementor Implementor is: Abstract Class or Interface Name of Reference Suffix of the Concrete Implementor NOTE The Next > button is activated. In the Deriver Classes screen, add or remove classes you want to use.
	Composite	Component Class Composite Class NOTE The Next > button is activated. In the Composite Operations screen, add or remove operations you want to use.
	Decorator	Component Class Decorator Class Concrete Decorator Class NOTE The Next > button is activated. In the Decorator Operations screen, add or remove operations you want to use.

Category	Design Pattern	Properties
	Observer	Subject Class Observer Class Concrete Subject Class Concrete Observer Class
	Proxy	Subject Class Proxy Class Real Subject Class NOTE The Next > button is activated. In the Proxy Operations screen, add or remove operations you want to use.
	Singleton	Singleton class
	Visitor	Visitor Class Implementation Style: visit (MyClass), visitMyClass (MyClass) NOTE The Next > button is activated. In the Elements to Visit screen, select classes and interfaces you want to visit.
Java Specific Java design patterns	Main	Main class
	RMI	Java RMI classes Remote Interface NOTE The Next > button is activated. In the Remote methods screen, select methods from the All list to the Selected list.

Category	Design Pattern	Properties
	EJB	
	Entity	Synchronize Names EJB Name EJB Class Remote Interface Home Interface Local Interface Local Home Interface Display Name Large Icon Small Icon Abstract Schema Name Cmp Version Persistence Type Reentrant Primary Key Class
	Message Driven	EJB Name EJB Class Display Name Large Icon Small Icon Acknowledge Mode Destination Type Subscribing Durability Message Selector Transaction Type

Category	Design Pattern	Properties
	Session	Synchronize Names EJB Name EJB Class Remote Interface Home Interface Local Interface Local Home Interface Display Name Large Icon Small Icon Session Type Transaction Type NOTE The Next > button is activated. In the Remote Methods screen, add or remove operations you want to use.
JUnit JUnit is a regression testing framework. It is used by the developer who implements unit tests in Java. JUnit is Open Source Software. The provided templates allow the user to create the constructions implemented in the JUnit framework. For more information, go to http://www.junit.org .	TestCase	TestCase Class Create suite() Create Constructor TestCase(String) Create runTest() Create setUp() Create tearDown()

Category	Design Pattern	Properties
	Tested Class	<p>Tested Class</p> <p>TestCase Class</p> <p>Create suite()</p> <p>Create Constructor TestCase(String)</p> <p>Create runTest()</p> <p>Create setUp()</p> <p>Create tearDown()</p> <p>NOTE: The Next > button is activated. In the Tested Operations pane, add or remove operations you want to use.</p>
XML Schema Specific XML design patterns	XSD complex Type	<p>Target Class</p> <p>Content: XSDcomplex content, XSDsimple content.</p>
	XSD compositor	<p>Target Class</p> <p>Compositor: XSDall, XSDchoice, XSDsequence</p> <p>Particle</p>
	XSD simpleType	<p>Target Class</p> <p>Content: XSDrestriction, XSDlist, XSDunion</p>
	XSD simpleType (XSDlist)	<p>Target Class</p> <p>Item Type</p>
	XSD simpleType (XSDunion)	<p>Target Class</p>

Category	Design Pattern	Properties
	Simple XSD restriction	Target Class Stereotype: XSDsimpleType, XSDsimpleContent Base Min Exclusive Max Exclusive Max Inclusive Min Inclusive Total Digits Fraction Digits Length Min Length Max Length White Space Pattern
WSDL Specific WSDL design pattern	Binding	Use the Binding pattern when you want to create binding of some PortType.
CORBA IDL Specific CORBA IDL design patterns	Interface	Name Abstract Local Interface
	Value Type	Name Abstract Custom Value Type
	Type Definition	Name Type Definition Specifier: typedef, boxed value Base Type

Category	Design Pattern	Properties
	Sequence	Name Base Type Sequence Size Anonymous
	Array	Name Base Type Array Size Anonymous
	Fixed	Name Digits Scale Anonymous
	Union	Name Discriminator Type
	Enumeration	Name
	Struct	Name
	Exception	Name
< Back		Go back to the Pattern screen.
Next >		Go to the other appropriated screen.
Finish		Finish and implement the wizard. The appropriated classes and interfaces are created.
Cancel		Cancel the wizard without implementing your actions.
Help		The MagicDraw Help is displayed.

Creating Setters / Getters

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

Setters and getters are common operations that contain almost every class. With the help of MagicDraw UML, set and get operations for class attributes can be generated automatically.

To create a setter or getter

- From the shortcut menu of the selected class, select **Tools**, and then select **Create Setters/Getters**. The **Select Attributes/Association Ends** dialog box opens.

- Add a tagged value "getter/setter for attribute=attribute_name" to the selected class.

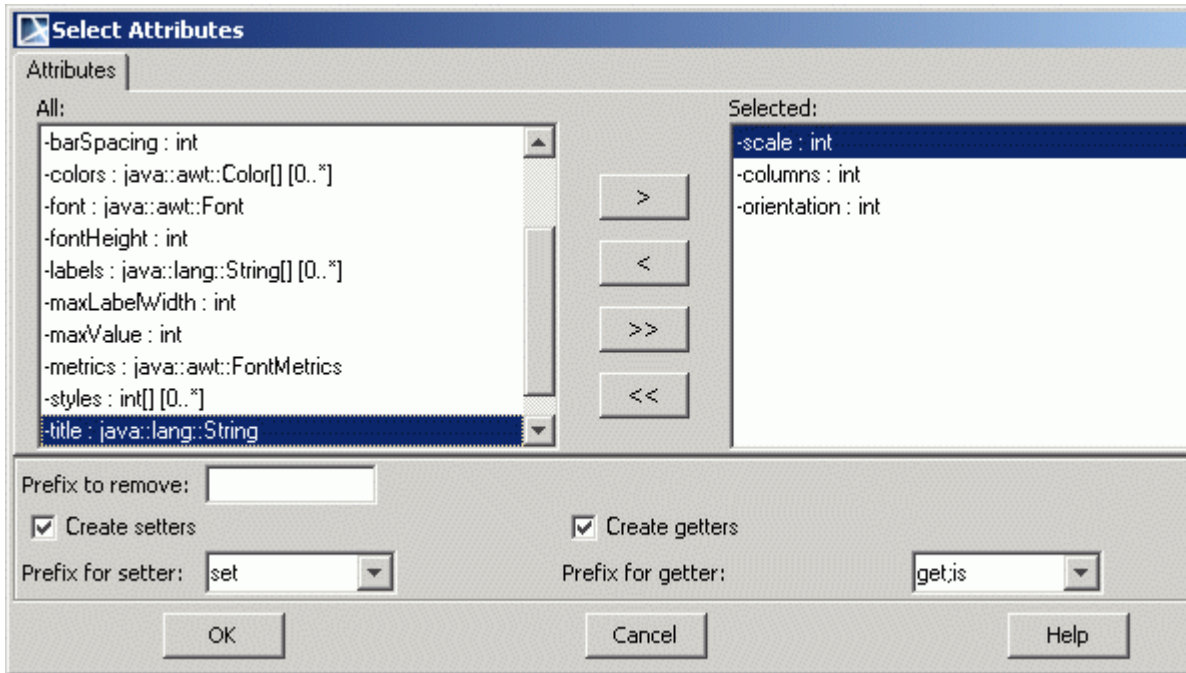


Figure 224 -- Select Attributes dialog box

Box	Function
All	Contains names of all attributes defined within the selected class.
Selected	Contains the selected attributes.
>	Moves the selected attribute from the All list to the Selected list. Setter for that attribute will be generated.
<	Moves selected attribute from the Selected list to the All list.
>>	Moves all attributes from the All list to the Selected list. Setters for all attributes will be generated.
<<	Moves all attributes from the Selected to the All list.
Prefix to Remove	Type a prefix of an attribute (-,) you want to remove while generating setters or getters.

Box	Function
Create Setters	Generates setters for the selected attributes.
Create Getters	Generates getters for the selected attributes.
Prefix for Setter	Select a prefix for the generated setter (operation). Possible choices: set or Set.
Prefix for Getter	Select a prefix for the generated getter (operation). Possible choices: get;is, Get;Is; get, or Get. NOTE "Get" is used for every getter, "is" is used if the type of an attribute is set as Boolean.
OK	Generates setters and/or getters for attributes that are in the Selected Items list.
Cancel	Exits the dialog box without any changes.
Help	Displays the MagicDraw Help.

The names of created operations (setters) are combined according to the following format:

```
public void set + <attribute name> (<attribute type> <attribute name>)
```

For example, if you have an attribute called x of type int, then the generated setter will look this way:

```
public void setx (int x)
```

The names of created operations (getters) are combined according to the following format:

```
public <attribute type> get + <attribute name> ( )
```

For example, if you have an attribute called x of type int, then the generated setter will look this way:

```
public int getx ();
```

Implementing or Overriding Operations

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

When you inherit classes from the base class that has abstract functions, you have to redefine them in the inherited classes. The implement/override operations tool will help you generate operations that are defined as abstract in the base class.

The **Implement/Override Operations** command can be invoked in 2 cases:

- When one classifier inherits operations from the base classifier (Generalization relationship).
- When some classifiers implement an Interface (Realization relationship).

To start the **Implement/Override Operations** tool

From the shortcut menu of the selected class, select **Tools**. Then, select **Implement/Override Operations**. The **Select Operations to Implement/Override** dialog box opens.

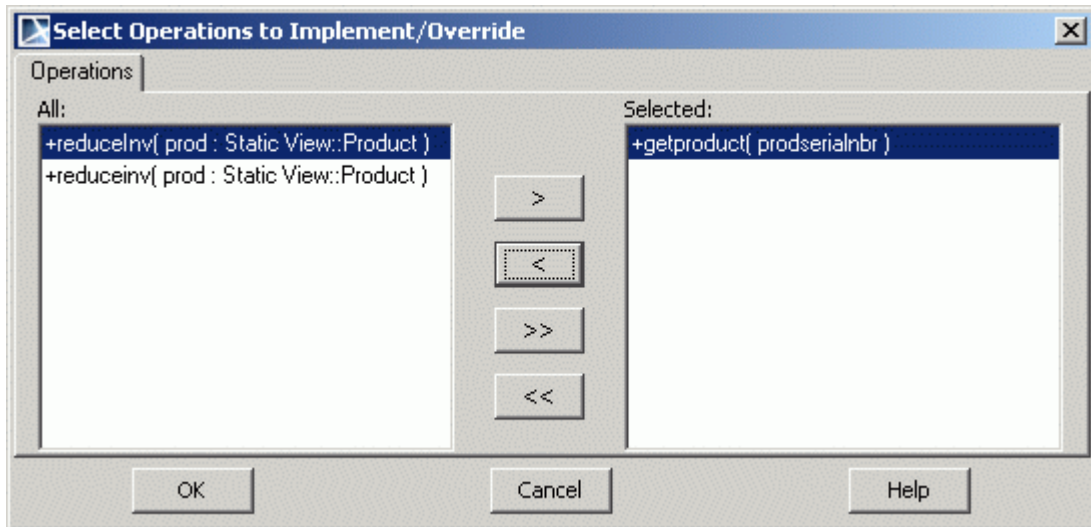


Figure 225 -- Select Operations to Implement/Override dialog box

Box	Function
All	Contains names of all operations defined within the selected class.
Selected	Contains the selected operations.
>	Moves the selected operation from the All list to the Selected list.
<	Moves selected operation from the Selected list to the All list.
>>	Moves all operations from the All list to the Selected list.
<<	Moves all operations from the Selected to the All list.
OK	Generates operations that are in the Selected list.
Cancel	Exits the dialog box without any changes.
Help	Displays the MagicDraw Help.

TIP!: Double-click the item name and it will be moved to the opposite list.

Model Transformation Wizard

[View Online Demo](#) Transformations**NOTE** This functionality is available in Architect and Enterprise editions only.

Model transformation is a data set that describes the action of applying model transformations to a set of packages. Components of model transformations include transformation types, parameters, packages set to transform, and the destination for the model transformation.

Types of transformations

Transformations can be categorized by the formats, models, and other entities they are being transferred *from* and *to*:

- **Any to Any** - copies all or part of your model to another package without making changes. You can also change the destination model by applying transformation mapping rules.
- **UML to XML Schema** - creates XML Schema models from pure UML models. Once the source model is copied, the necessary stereotypes are applied according to the XML schema modeling rules.
- **XML Schema to UML** - creates a pure UML model from the XML Schema. The source model is copied, but stripped of all unnecessary stereotypes. Because it derives abstract models from more specific ones, reverse transformation can occasionally lose information in the destination model.
- **Profile Migration** - transformation provides a way to migrate your model according specified mapping rules.
- **Generic DDL to Oracle DDL** - creates more specific Oracle DDL models from Generic DDL. In this transformation, mapping will be performed using appropriate Oracle DDL types.
- **UML to Generic DDL** - creates generic DDL models from UML models. Once the source elements are copied to the destination, the appropriate stereotypes are applied. This transformation works only with classes, packages, and association relationships.
- **UML to Oracle DDL** - creates Oracle DDL models from UML models. Mapping is performed using the appropriate Oracle DDL types.

- **DDL to UML** - derives UML models from DDL models.

Model Transformation Wizard

To start the **Model Transformation Wizard**

- From the **Tools** main menu, select **Model Transformations**.
- Select the package(s). From the shortcut menu, select **Tools** and then select **Transform**.

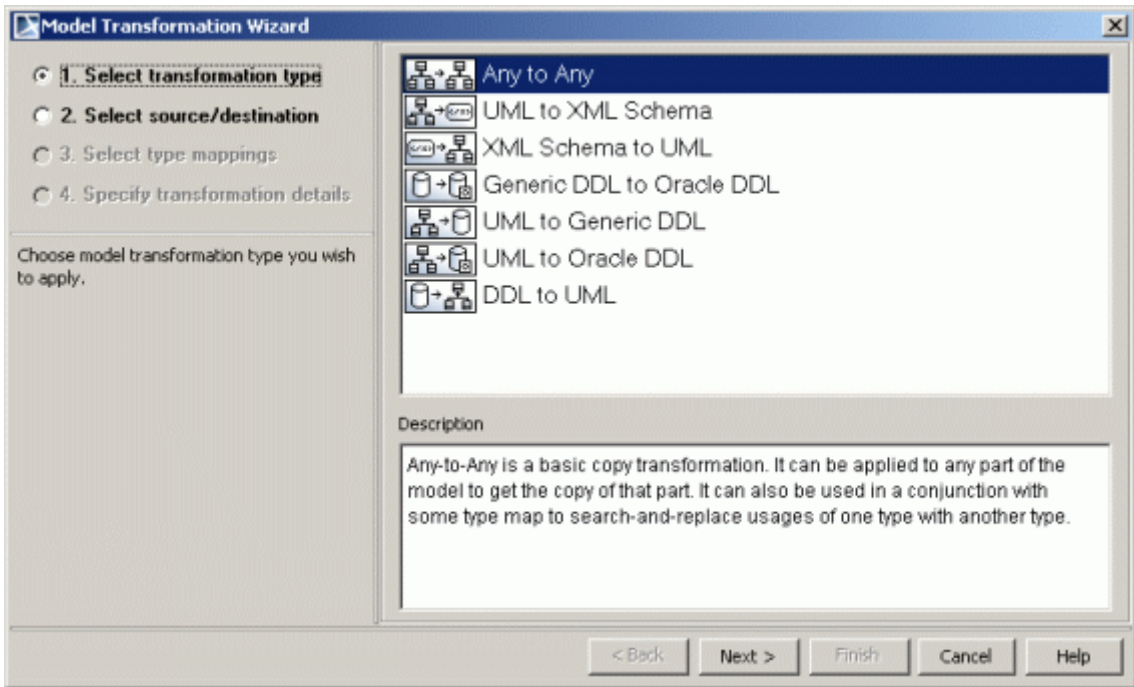


Figure 226 -- Model Transformation Wizard. Select transformation type

Transformation types are displayed in the list window.

The following operations are available in the **Select transformation type** window:

- **Next >** – proceed to the next step (in this case, Select source/destination.)

- **Cancel** – cancel the wizard.
- **Help** – display the MagicDraw Help.

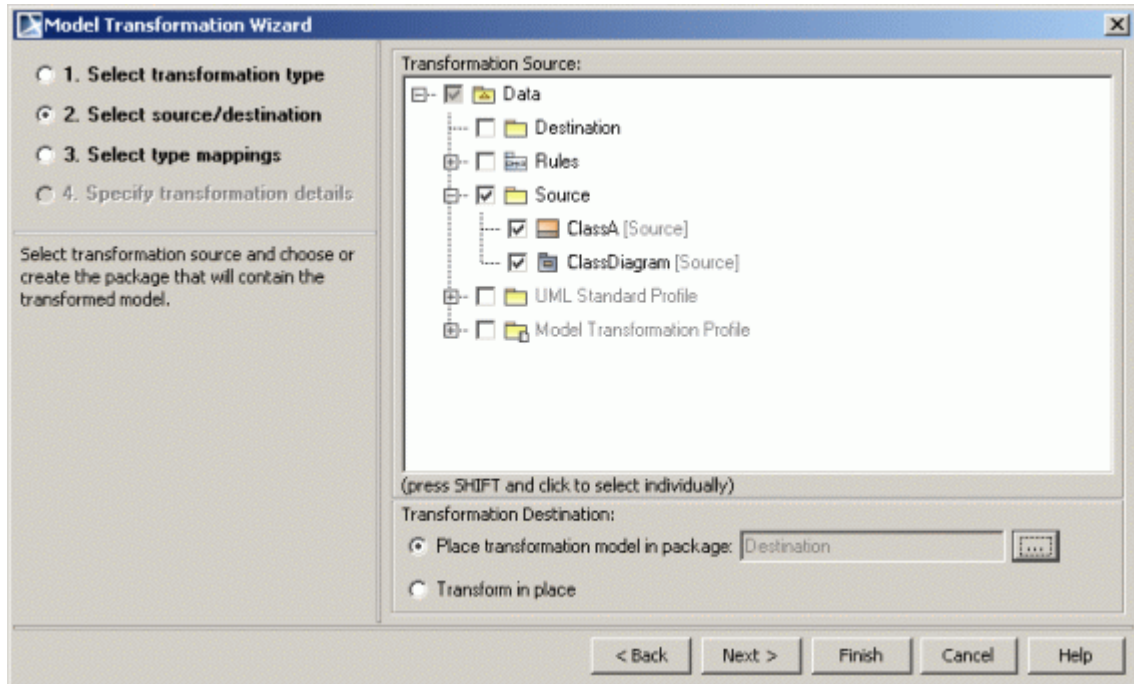


Figure 227 -- Model Transformation Wizard. Select source/destination

The **Transformation Source** tree displays all project data (the packages and their inner elements) that can be selected as sources for transformation.

Select the **Place transformation model in package** option button to specify the package to which the source will be transformed. Click the “...” button to display the **Destination Package** dialog box. Select an existing package from the **Packages** tree or create a new one.

Select the **Transform in place** option button if you want the source package structure to correspond to the destination package.

The following operations are available in the **Select source/destination** window:

- **< Back** – return to the previous dialog box.
- **Next >** – proceed to the next step (in this case, Select type mappings.)

- **Finish** – finish the transformation configuration. All other options will be set by default. The **Model Transformations Wizard** exits and transformation results appear in the project.
- **Cancel** – cancel the wizard.
- **Help** – display the MagicDraw Help.

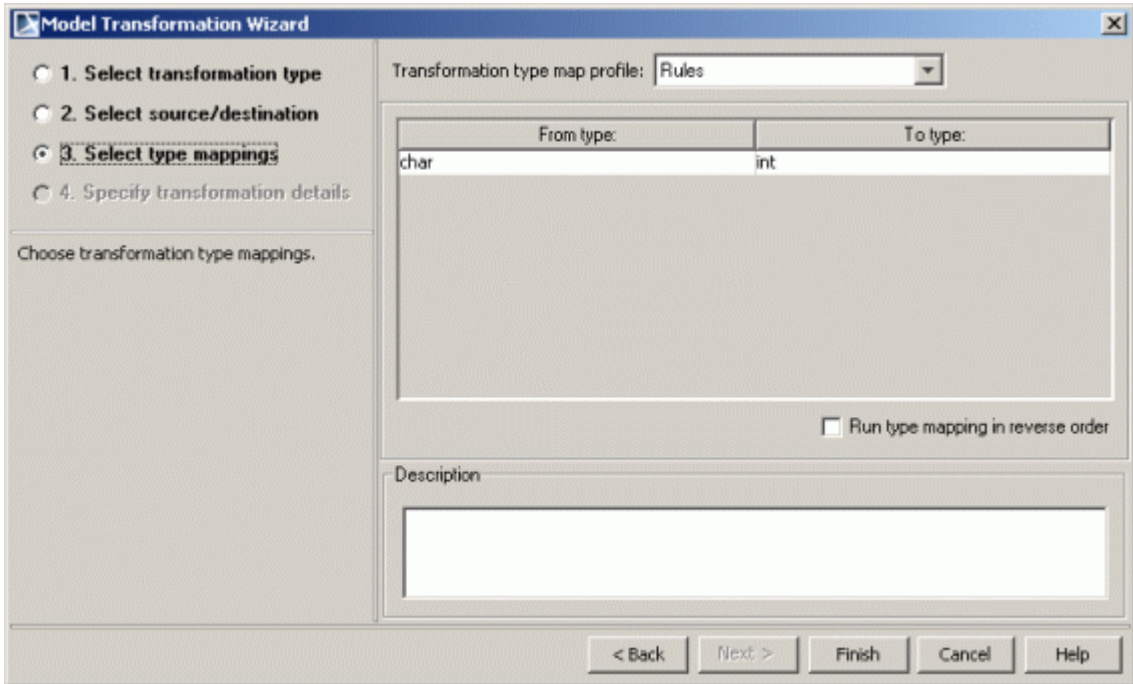


Figure 228 -- Model Transformation Wizard. Select type mappings

To see a list of the default or manually created mapping profiles, click the down arrow in the **Transformation type map** field. These mapping profiles specify which mapping rules should be applied to the model during transformation.

The **From type** and **To type** fields display the type mappings of the selected transformation profile.

The **Run type mapping in reverse order** check box creates opposite type mapping in transformations.

The following operations are available in the **Select type mapping** window:

- **< Back** – return to the previous dialog box.

- **Next >** – proceed to the next step - Specify transformation details. (This button is disabled during **Any to Any** transformations.)
- **Finish** – finish the configuration of the transformation. The **Model Transformations Wizard** exits and transformation results appear in the project.
- **Cancel** – cancel the wizard.
- **Help** – display the MagicDraw Help.

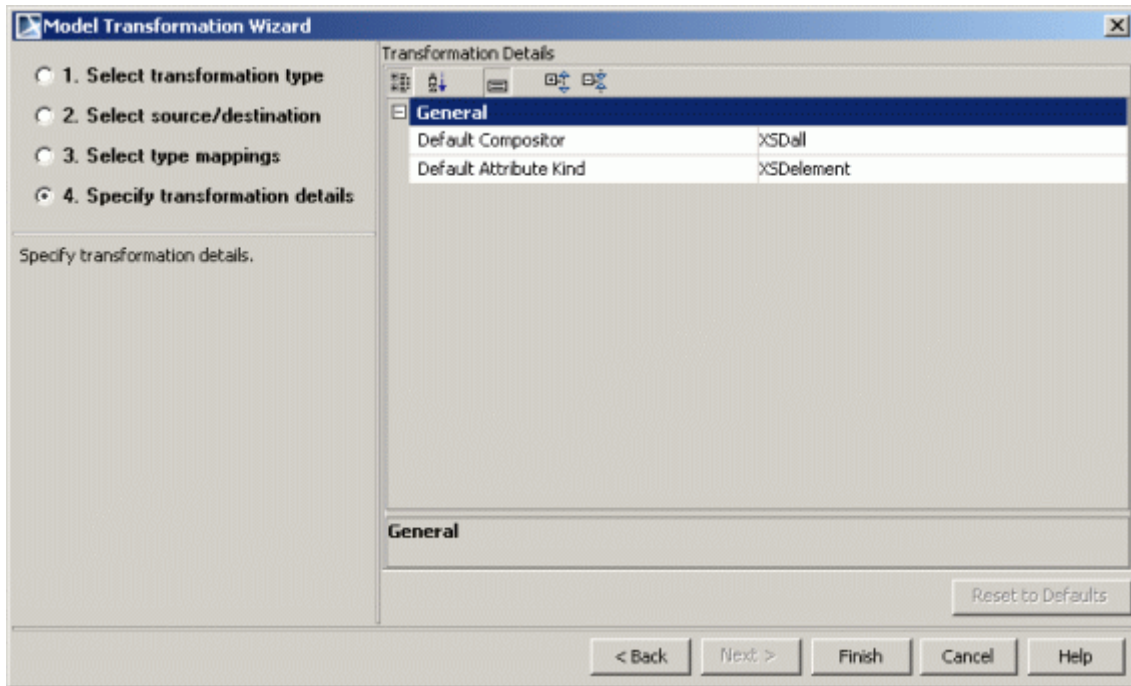


Figure 229 -- Model Transformation Wizard. Specify transformation details

The **Transformation Details** table displays the various properties (selected in Step 1 of the **Model Transformation Wizard** of a specific transformation. Here you can change the element's default value.

Click **Reset to Defaults** to change the transformation properties to the default values.

The following operations are available in the **Specify transformation details** window:

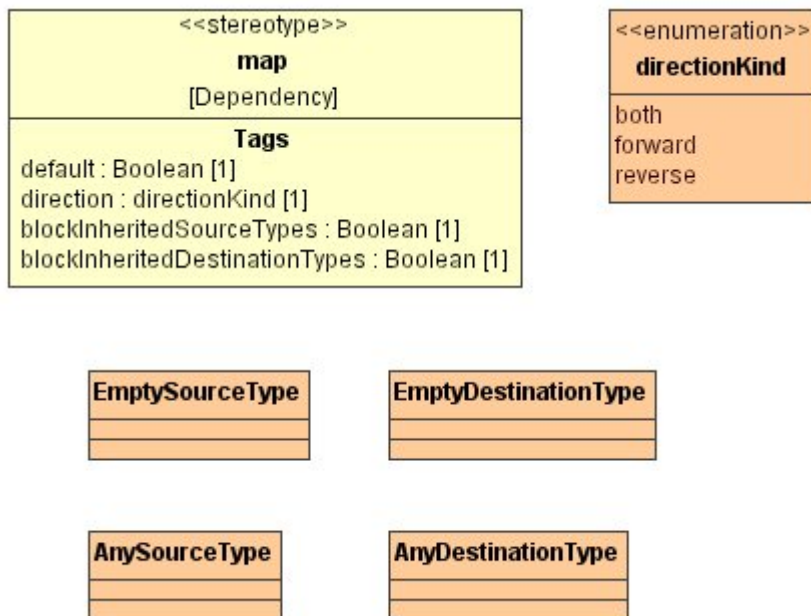
- **< Back** – return to the previous dialog box.

- **Finish** – finish the transformation configuration. The **Model Transformations Wizard** exits and transformation results appear in the project.
- **Cancel** – cancel the wizard.
- **Help** – display the MagicDraw Help.

Model Transformation Mapping

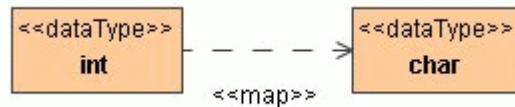
When transforming your model, you can map the types used in the transformation classes. This mapping is usually used to map primitive types from one domain to appropriate primitive types in another domain.

The map is a collection of rules framed by the "from type - to type" relationship. The following elements can be used in creating mapping rules. They are stored in the *Model Transformation Profile*.



To create transformation mapping rules

1. Import or use *Model_Transformation_Profile.xml.zip*.
2. Apply the stereotype named `<<typeMap>>` to the newly created rules package.
3. Select the standard types that will be used in transformation mapping, then create the desired dependency relationship between the two types. Apply the stereotype named `<<map>>` to the dependency.



In the example above, after the transformation, all *int* types will be transformed to *char*.

- NOTE:** To change the behavior of the mapping, See “Transformation Mapping Options” on page 486.
4. Start the **Model Transformations Wizard**. In Step 3, select type mapping, click the **Transformation type map** profile field and select the package containing mapping rules.
 5. Click **Finish**. Transformation mapping, as specified by the newly-created rules, will be performed on the transformed model.

Transformation Mapping Options

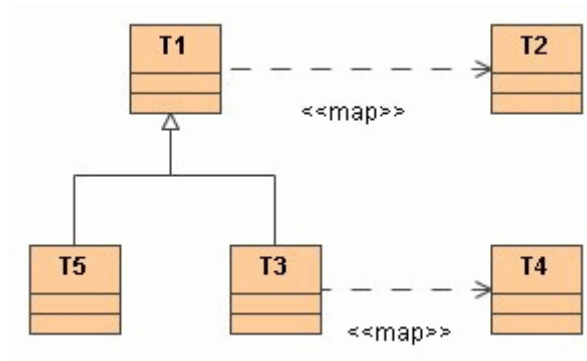
To change a mapping behavior, set appropriate tags on the dependency relationship:

- Default type mapping can be applied both ways. For example, you can use the type map from *int* to *char* to change all *int* types to *char*. But if you select the **Run type mapping in reverse order** check box under the **Select type mappings** (Step 3 of the **Model Transformation Wizard**), the type map will be applied in reverse order - all *char* will be changed to *int*. To change this, create a value for the **direction** tag in the **Dependency Specification** dialog box, **Tags** tab.

The default value is "both". Change this value to "forward" to make this mapping valid for forward dependencies. Change the value to "reverse" to have this mapping valid for reverse dependencies.

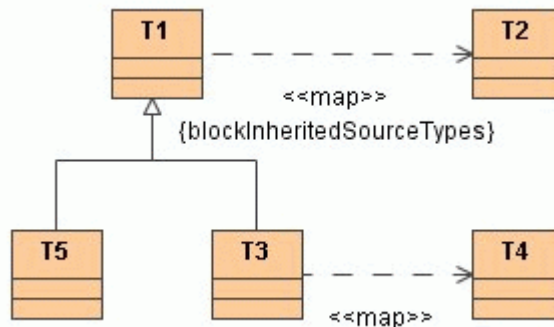
- To apply the order of direction for all dependencies, set the **defaultDirection** tag value in the **Package Specification** dialog box. This package contains the `<<typeMap>>` stereotype. Initially, the `<<map>>` dependency **direction** tag is checked. If not found, the `<<typeMap>>` package **defaultDirection** tag value is set.

- You may have several `<<map>>` relationships from one source or to one destination. In these cases, the "default" tag for at least one of the relationships must be set. Dependencies from one source without **default** tags (or with several) is invalid. Also, dependencies to one destination with "reverse" or "both" direction order set, must have one with a **default** tag.
- Transitive mapping for the elements (for example, Type1 -> Type2 -> Type3) is not supported.
- You may also control mapping behavior for the type inheritance source of the model. By default, derived subtypes are also mapped by the rule governing the parent type (unless, of course, they have their own rules for mapping). If the `blockInheritedSourceTypes` tagged value is set, derived types are not affected by this rule.



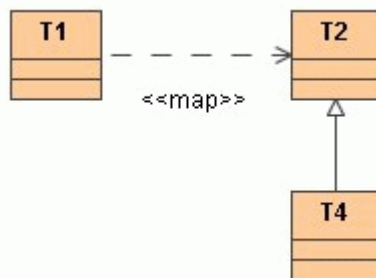
Here T1, T2, and T5 are types in the source domain, while T2 and T4 are types in the destination domain. Given these two mappings (T1 -> T2 and T3 -> T4), the following statement is true: T1, and all types derived from it (such as T5), are mapped to the T2 type, except T3 and any of the types derived from it. These types are mapped to T4.

Now consider an example where `blockInheritedSourceTypes` is set:



In this case, T3, along with the types derived from it, are still mapped to T4. T1 is still mapped to T2. However, unlike the previous example, T5 and all the types inherited from T1 are NOT mapped to T2.

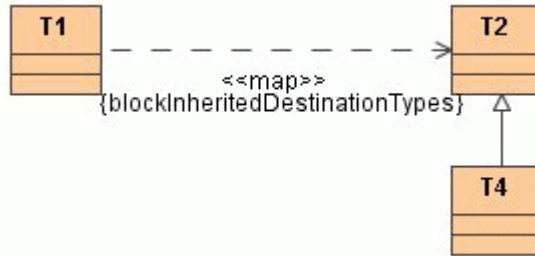
- You can also control the mapping behavior of the type inheritance in the destination model. This is only effective on the second (and successive) applications of the transformation. By default, derived subtypes in the destination model are not overwritten since they are considered suitable substitutes of their parent.



Here T1 is a type in the source domain, while T2 and T4 are types in the destination domain. Given this mapping (T1 ->T2), on the first application of the transformation, type T1 residing in the source model will be mapped to type T2 in the destination model.

Now let us look at a case where the user refines the destination model - changing the type on the destination model attribute from T2 to - T4. This situation is quite common: for example, the user refines an attribute type from string to basic URI in the XML schema, or from Integer to nonNegativeInteger, and so forth). In essence, the mapping for inherited types of T2 is performed as if there was a mapping T1->T2 (default), T1->T4, T1-> any_other_type_inherited_from_T2.

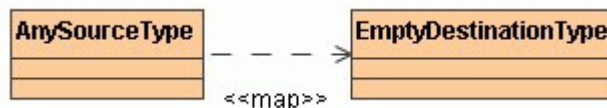
Now consider what happens when we apply the **blockInheritedDestinationTypes** tagged value:



In this case, type T4 has no special treatment. If the user applies the transformation, T1 is mapped to T2. In another scenario, the user refines the destination model, changing the attribute type from T2 to T4. If the user now reapplies (updates the destination) the transformation, the attribute type is overwritten: T4 is reset back to T2.

When the user loads the type map in the reverse direction, the roles of the "blockInheritedSourceTypes" and "blockInheritedDestinationTypes" are transposed (unless of course the "direction" mandates that this mapping is not used in the reverse direction).

- The special type **EmptySourceType** is used to indicate that the attributes with no type should be mapped with this dependency.
- The special type **EmptyDestinationType** is used to indicate that the attributes in the destination classes should have no type after remapping (type removal).
- The special type **AnySourceType** - is a template that matches any type in the source model (see mapping rules for type inheritance). By using this type, together with the inheritance mapping rules, the user can specify that any other types not defined by the mapping should be interpreted by this mapping. According to this rule, any other types in the source model should be stripped in the destination model.



- The special type AnyDestinationType is a template that matches any type in the destination model (see mapping rules for type inheritance).

Model Transformation Update

When reapplying a transformation, relationships are retrieved from storage. They are then used to find the model elements in the destination model that correspond to the given model element in the source model.

The presence of unmapped model elements in the source model indicates that these are newly added elements. Apply the usual rules for the transformation using the same behavior as if this were the first application of the transformation.

If the model elements already contain mapping, then the user can either update properties of the mapped element or leave the transformation model intact.

To update a transformed model

- From the destination package shortcut menu, select **Tools** and then select **Update Transformed Model**. The **Model Refresh Options** dialog box opens:

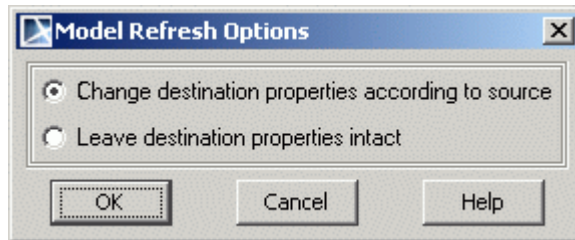


Figure 230 -- Model refresh options dialog box

The **Change destination properties according to source** option overwrites element properties in the destination model with properties from the source model (but only the elements connected with mapping dependencies).

The **Leave destination properties intact** option does not overwrite properties in the destination model with properties from the source model (retaining changes made to the destination model while ignoring changes made to the source model).

NOTE This update option works only with data for elements that have already been created (renaming, changing type, etc.) and contain type mapping dependencies. If new elements are added to the source, once the update occurs, copies of the new elements will be created in the destination model. If an element is deleted from the source, it will not be removed from the destination after the update.

Profile Migration Transformation

Transformation provides a way to migrate your model according specified mapping rules.

Profile migration transformation search-and-replaces the following elements:

- applied stereotypes,
- tagged values,
- usages of one type with another type.

Element symbols are updated according to model changes.

Before starting transformation you have to define transformation mapping.

Profile Migration Transformation mapping

Before starting transformation you have to create transformation mapping rules.

To create mapping rules you need to create Dependency relationship between elements you want to transform. Mapping rules can be created (dependency relationship can be created) between the following elements:

1. Stereotypes,
2. Tags,
3. Types.

1. Creating mapping rules for Stereotype transformation

This type of transformation is used to replace Stereotype. To create Stereotype transformation mapping rule:

1. Create Dependency relationship between Stereotypes which you want to transform.
2. Apply *ReplaceStereotype* stereotype to Dependency.
3. Perform transformation.

Tag values of old stereotype are preserved when tag name and type of tag value is the same. For tag values with different names create mapping rule for tag transformation.

ReplaceStereotype stereotype has the following tags:

- **disableNewTypeCreation** tag. By default false. Set this tag to true if you do not want to perform transformation when target and source metaclasses are not compatible. For example, if you do not want that Class would be changed to Use Case.
- **disableReplaceWhereSaveAsElementValue** tag. By default false. Set this tag to true if you do not want that stereotype would be changed where it is used as Tag value (tag value is stereotype itself, for which you perform transformation).

2. Creating mapping rules for Tag transformation

This type of transformation is used to replace Tag (when tag names differs). For example, source stereotype has *author* Tag and target stereotype has *name* Tag.

To create Tag transformation mapping rule:

1. Create Dependency relationship between Tags which you want to transform.
2. Apply *ReplaceTaggedValue* stereotype to Dependency.
3. Perform transformation.

NOTE: To create mapping rule correctly, you have to create Dependency relationship not only between Tags, but also between Stereotypes of these tags (see mapping between stereotypes, which is described above).

3. Creating mapping rules for types transformation

This type of transformation is used to replace type. For example, to replace type of Attribute.

To create types transformation mapping rule:

1. Create Dependency relationship between Types which you want to transform.
2. Apply *ReplaceType* stereotype to Dependency.
3. Perform transformation.

Starting Profile Migration Transformation

To start the **Model Transformation Wizard**:

- From the **Tools** main menu, select **Model Transformations**.

Or

- Select the package(s). From the shortcut menu, select **Tools** and then select **Transform**.

To start the Profile Migration Transformation:

- In the Model Transformation Wizard select the **Profile Migration** transformation.

Note: The **Next** step in wizard will be disabled if there will be no defined mapping.

Sample of the Profile Migration Transformation

This sample describes step-by-step instructions how to create profile migration mapping rules and perform transformation. In this sample we will change one stereotype to other.

1. Create *Book* stereotype with Class metaclass.
2. Create your model, for example, create *Source* package with *Source* Class diagram.
Draw *Sample* Class and apply *Book* stereotype.
To change *Book* stereotype to other, for example to *Magazine* stereotype, you have to create profile migration transformation mapping rules. Follow next steps for creating mapping.
3. Create stereotype *Magazine* with Class metaclass.
4. From the *Magazine* stereotype to *Book* stereotype draw Dependency relationship.
5. To the Dependency relationship apply *ReplaceStereotype* stereotype (see Figure 1).
Profile Migration transformation mapping rule is created. Now you can start transformation.
6. To start transformation select **Model Transformations** from the **Tools** menu. The **Model Transformation Wizard** will open.

7. Select the **Profile Migration** transformation and click **Next**.
8. Select the *Source* package in the *Select source/destination* step (Figure 3). Click **Next**.
9. The **From** and **To** fields display the mappings of the selected transformation profile in the *Check mappings* step (Figure 4). Click **Finish**.

After this transformation stereotype of *Sample Class* will be changed to *Magazine* stereotype.

NOTE: For more information about Model Transformation Wizard, see “Model Transformation Wizard” on page 481.

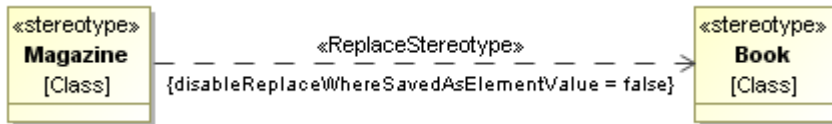


Figure 231 -- Profile Migration Transformation mapping

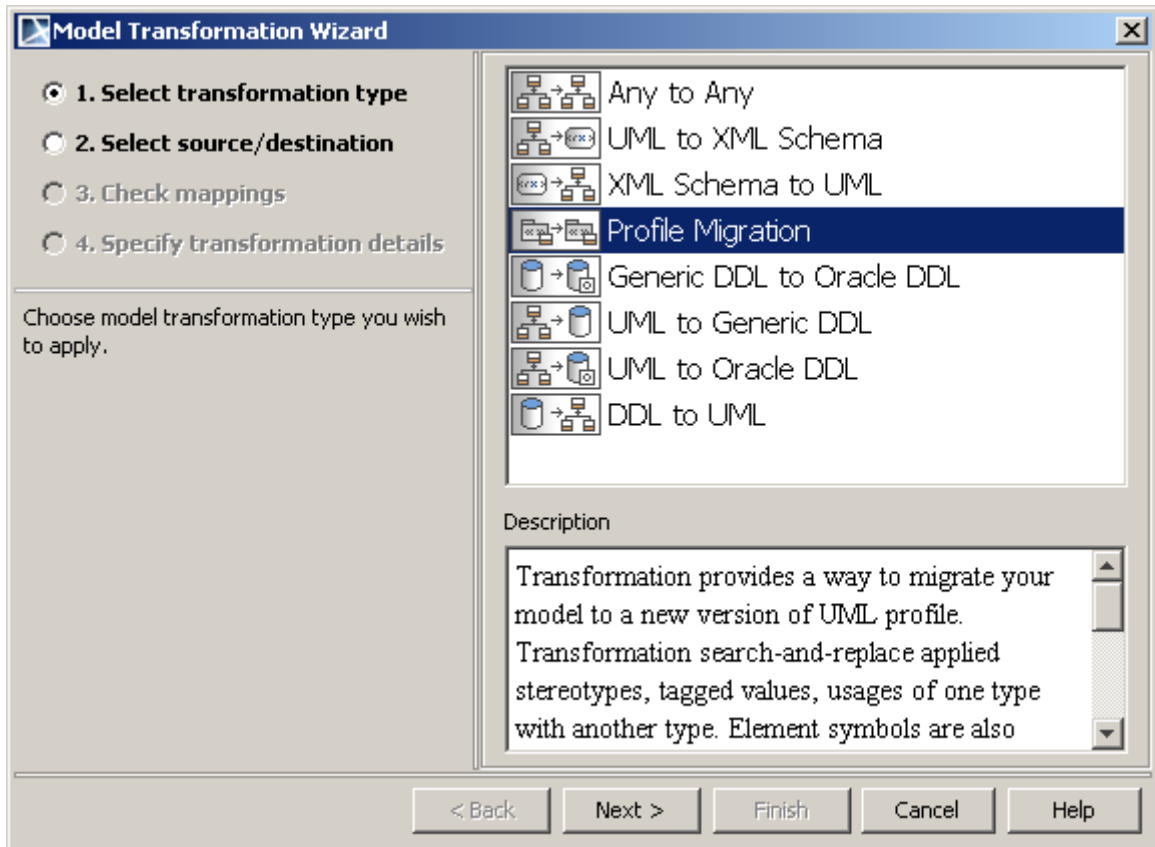


Figure 232 -- The Model Transformation Wizard, the Select transformation type step

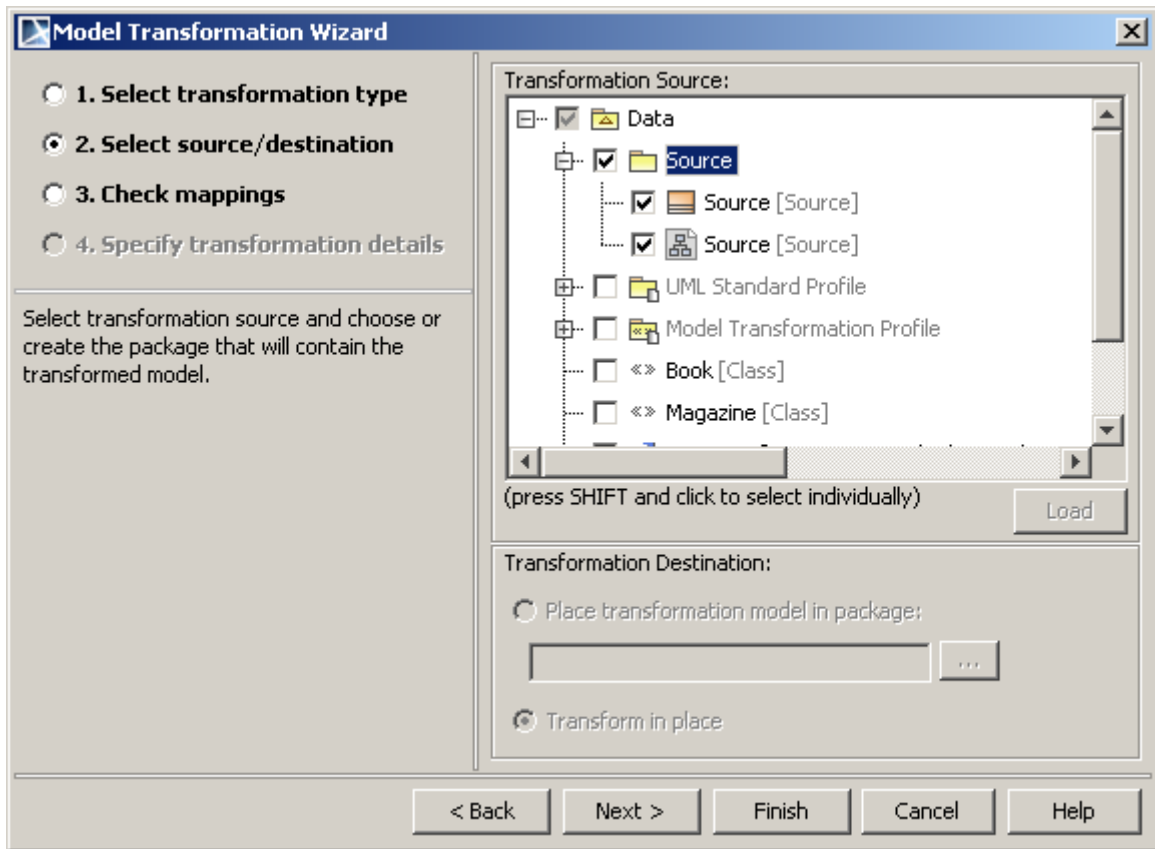


Figure 233 -- The Model Transformation Wizard, the Select source/destination step

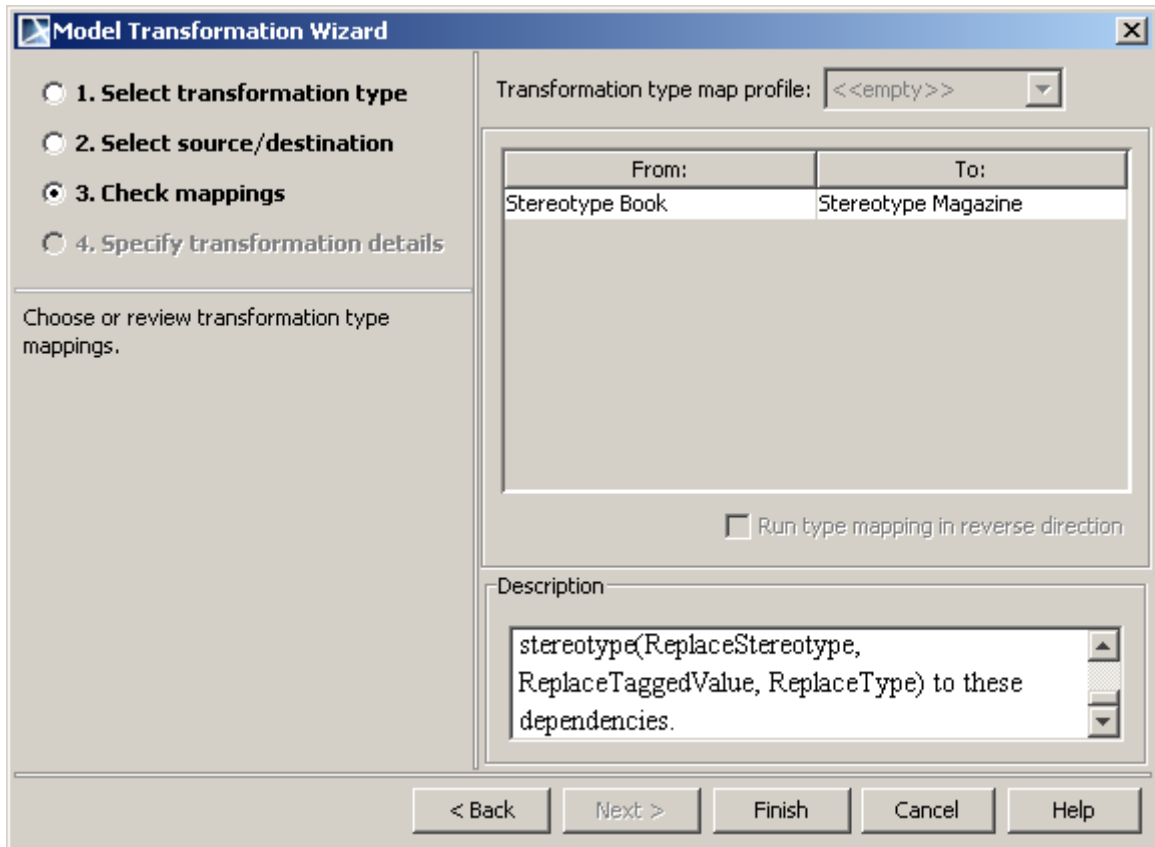


Figure 234 -- The Model Transformation Wizard, the Check mappings step

Resource Manager

MagicDraw Resource Manager functionality allows you to manage local resources (installed with MagicDraw, downloaded) and resources available on the web.

With RM (Resource Manager) you can manage different types of resources (Profiles, Plugins, Templates, Language resources, Case studies/examples, Custom diagrams, and others).

RM allows:

- Review
- Remove (delete)
- Import, download
- Update resources.

The following are benefits of the Resource Manager functionality:

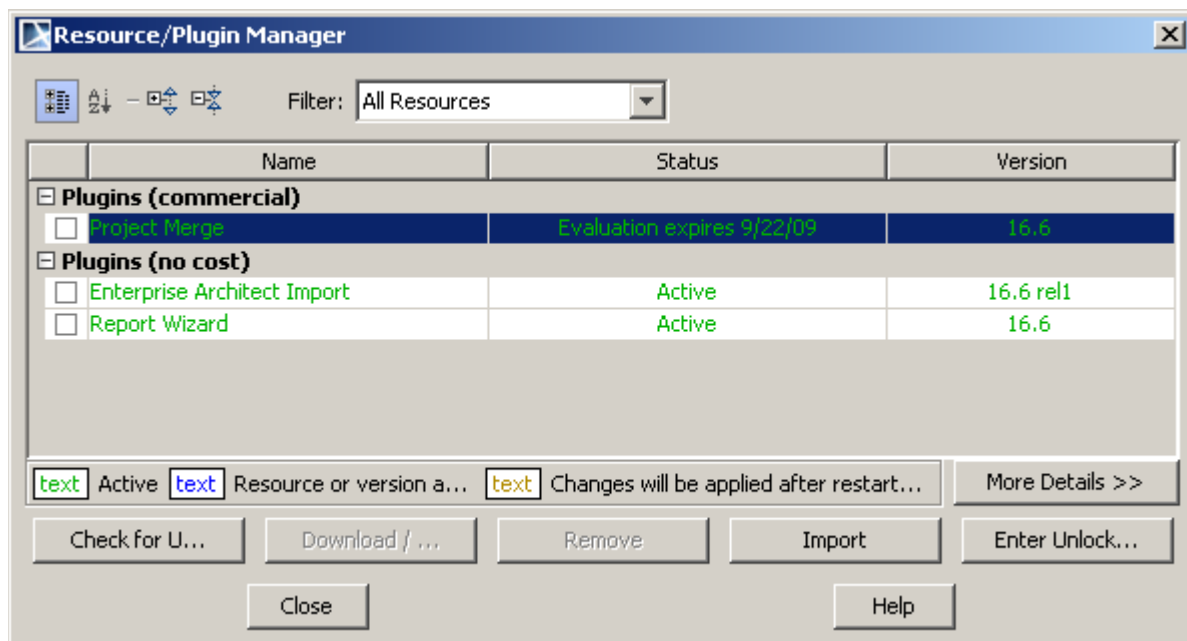
Benefits:

- Easier to find needed resources and download them.
- More informative resource descriptions.
- The possibility to create your own resources and share them.
- The possibility to check resource dependency in the required table.

The Resource Manager functionality is included in all MagicDraw editions, except, Reader.

To open the Resource Manager:

From the [Help main menu](#), select the **Resource/Plugin Manager** command. The [Resource / Plugin Manager window](#) opens.



The table below lists items of the Resource Manager window.

Element Name	Function
Name	Different types of resources are listed in separate nodes. There are the following resources types: examples, templates, languages, and profiles. In front of each resource is a check box. Select this check box to manage this resource. If the resource is unavailable, it is impossible to select the check box in front of it (checkbox is grey). The required table rows contain information about why it is not available.
ID	Resource ID.
Category	Resource type.

State	<p>The following states of resources are possible:</p> <ul style="list-style-type: none"> • Not installed (downloaded) • Will be installed after restart • Installed • Will be removed after restart • Not downloaded • removed (if resource does not exist on the web).
Ver Obtained	The Version Obtained column indicates the version for a resources and the state: installed, not installed, and removed.
Ver Available	<p>The Version Available Column indicates the newest resource version on the web.</p> <ul style="list-style-type: none"> • Does not exist on the web. If the resource does not exist on the web. • Check for updates. If the resource list has not been downloaded from the web.
Date	Resource creation date.
Size	Size of the particular resource.
Meaning of text colors	<p>One row lists the meanings of the text colors:</p> <ul style="list-style-type: none"> • Green - installed resources are marked green. • Blue - not downloaded resources are marked blue. If the resources are already installed or downloaded and the newer version is available, the resources are marked blue. • Black - changes will be fully applied after restarting MagicDraw.
More>>/<<Less	<p>Expands / Collapses the main Resource Manager window with additional fields.</p> <p>The following fields list all data of the selection in the resource list:</p> <ul style="list-style-type: none"> • Name, Resource home page, Provider, Description • Required table (with Name, Required, Status columns). If a resource is installed but a newer version is available on the web, the required table shows the newest version requirements.
Check for Updates	The Check for Updates button is inactive if the resource list has already been successfully downloaded.
Download/Install	Downloads the selected resources and installs. The Download/Install button is inactive if no resources have been selected.

Remove	Button is active if at least one resource is selected with states "not installed" or "installed". The resources of other states may not be removed from the RM list. The remove functionality is not available for required resources.
Import	The Open dialog is opened. Import the resource you need.

NOTE If the resource requires payment, the **Price** of the particular resource is displayed in the **Resource Manager** window under the **Price** column.

Spelling Checker

Spelling Checker enables you to:

- Check spelling as you type. A shortcut menu provides spelling options. Right click the word underlined in red to enter the shortcut menu. Spelling options will be displayed. Words can also be entered into a customized dictionary using **Add to Dictionary** (see "Spell checking as you type" on page 501).
- Check the spelling of a whole project or of a selected part. You can list all the spelling errors found in a project and correct them easily (see "Spell checking for the whole project or the selected scope" on page 505).
- Set Spelling Checker options. You can set spelling checker options, such as skipping numbers, upper case words in the **Environment Options** dialog box (**Spell Check** option) (see "Setting the spell checking options" on page 511).
- Add a spell checking dictionaries. All "Open Office" supportive spelling languages can be added additionally to the existing ones (see "Spell checking dictionaries" on page 513).

Spell checking as you type

On typing spell checker checks if typed word is correct. In case the word is incorrect it is underlined with red winding line. Right-click on underlined word invokes context menu with suggested editions and capability to add word to dictionary.

Spell checking as you type is performed in the following locations in MagicDraw:

- symbol names on diagram pane,

- for properties in the specification dialog boxes,
- in the documentation pane,
- in the Containment tree,
- In various location on log messages, names, typing and comment fields.

Right-click on underlined word to invoke the shortcut menu in the following MagicDraw window locations:

1. Diagram pane (see Figure 235 on page 502).
2. Containment tree (see Figure 236 on page 503).
3. Dialog boxes (see Figure 237 on page 504).

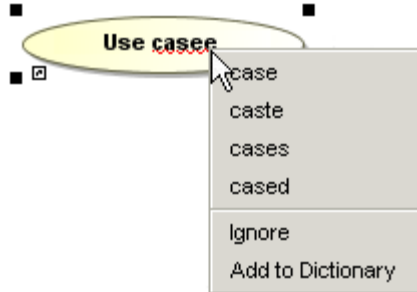


Figure 235 -- Shortcut menu of the spelling error on the diagram pane

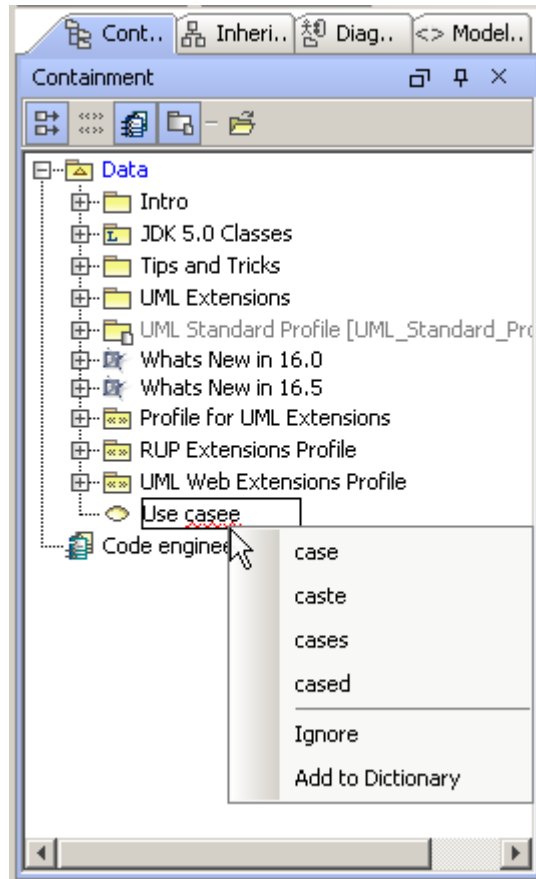


Figure 236 -- Shortcut menu of the spelling error in the Containment tree

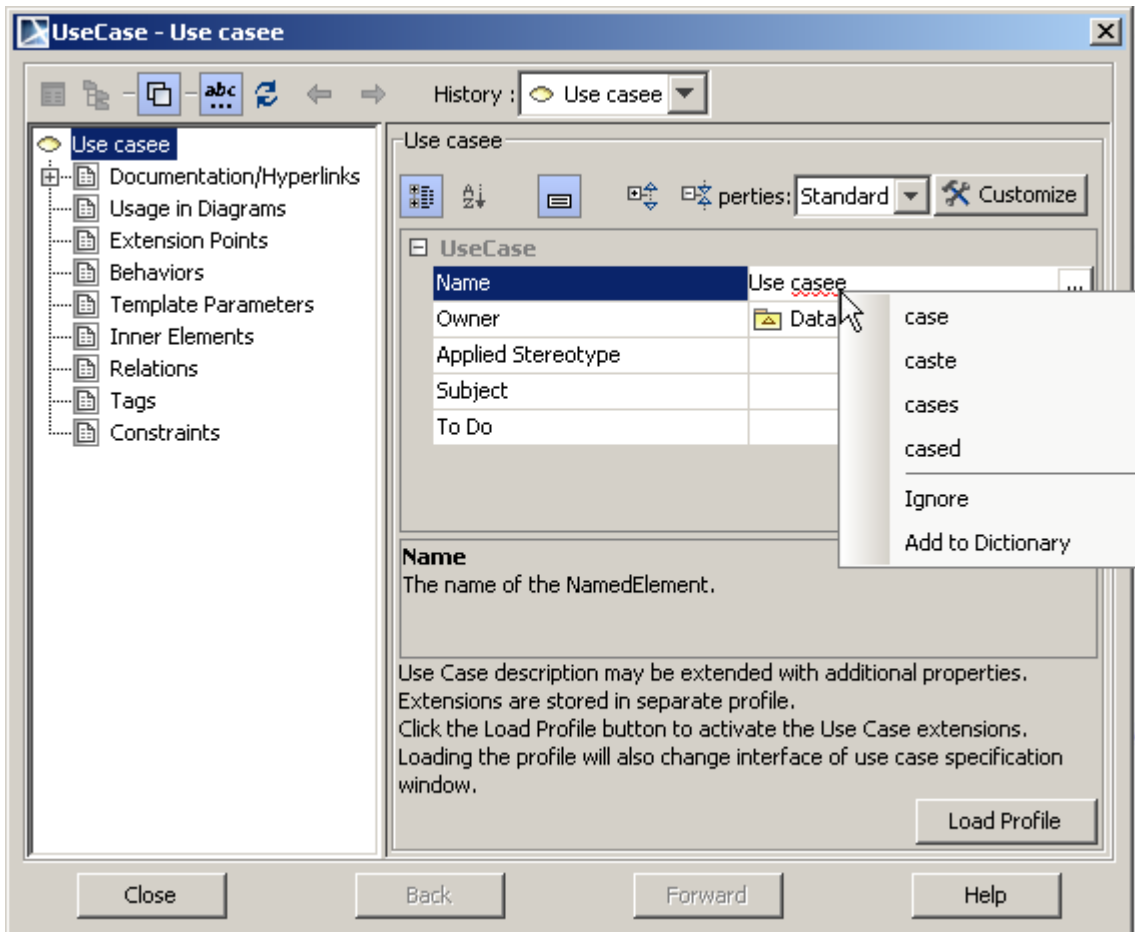


Figure 237 -- Shortcut menu of the spelling error in the dialog box

There are three ways to correct the spelling error:


1. Wrongly typed word can be changed by typing or by selecting provided suggestion that is always correct syntactically, but not always correct semantically.
2. The **Ignore** command. Select the **Ignore** menu item and the wrongly spelled word is treated as correct for this time, but will be discovered as wrongly spelled in the other case.

3. The **Add to dictionary** command. Select the **Add to dictionary** menu item in pop-up menu. By pressing this item the wrongly spelled word will be added to custom dictionary. Next time it will not be treated as wrongly typed.

NOTE: Wrongly spelled words are underlined only in edit mode of particular component. If edit mode has been left, underline disappears and vice versa.

Spell checking for the whole project or the selected scope

Checking spelling for the whole project

1. From the **Tools** main menu, select the **Check Spelling** command or press the **Check Spelling**  button in the main toolbar. The **Check Spelling** dialog box opens.
2. Click the **Check** button. The **Validation Results** dialog box opens.

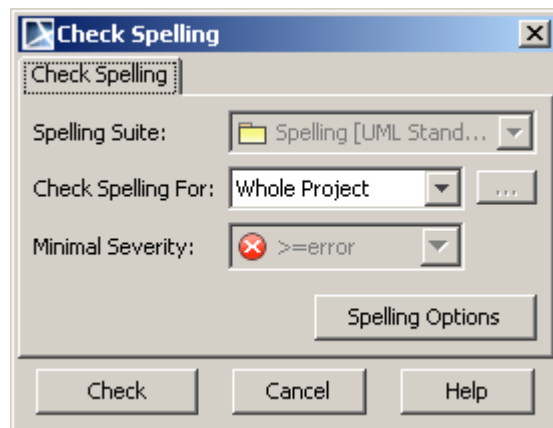


Figure 238 -- The Check Spelling dialog box

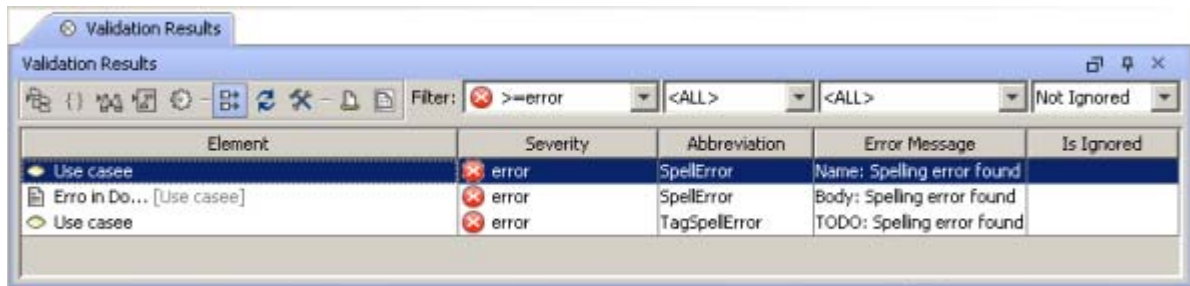


Figure 239 -- The Validation Results dialog box

Checking spelling for the selected scope

1. From the **Tools** main menu, select the **Check Spelling** command. The **Check Spelling** dialog box opens.
2. In the **Check Spelling For** combo box select the **Validation Selection** option.
3. Click the "... " button. The **Select Elements** dialog box opens. Select the scope to check spelling.
4. Click **OK**.
5. In the **Check Spelling** dialog box, click **Check**. The **Validation Results** dialog box opens.

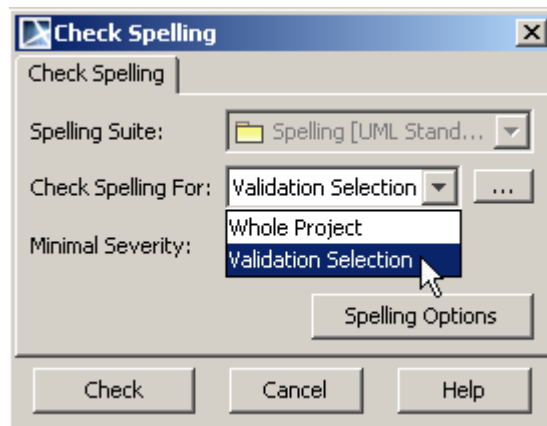


Figure 240 -- The Check Spelling dialog box, the Check Spelling For option

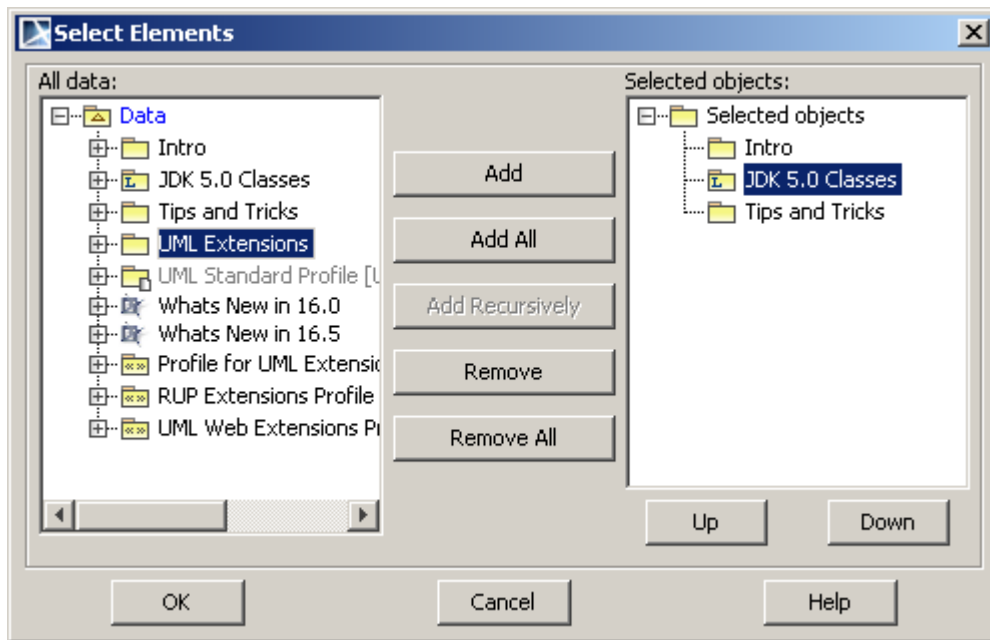


Figure 241 -- The Select Elements dialog box

Analyzing the Check Spelling (the Validation Results window)

The **Validation Results** window provides with all spelling errors. You can choose either to correct particular error or to ignore it. For more information about correcting the spell checking error, “Solving the spell checking errors” on page 508.

Spelling error in the **Validation Results** window has description of construct as follows:

Column title	Description
Element	Element with contains spelling error. Spelling error can be found in the element name, or inner properties, such as documentation, the To Do property and others.
Error Message	<Element property name>: <Spelling error found>

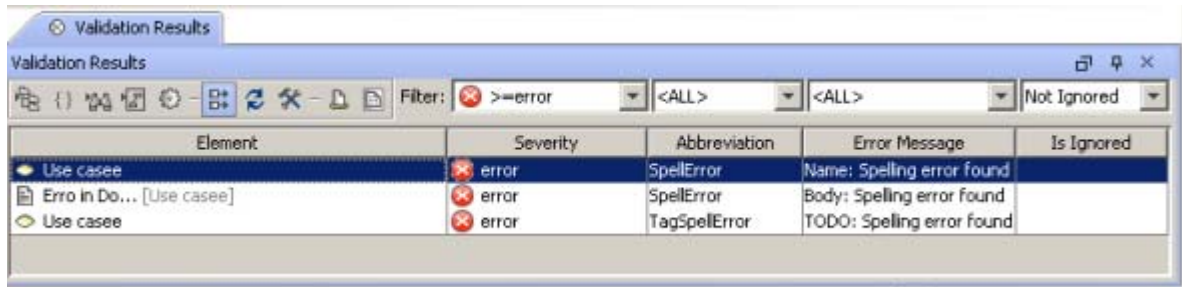


Figure 242 -- The Validation Results window

For more information about validation functionality, see “Validation” on page 601. Here the **Validation Results** window is described in more details.

Solving the spell checking errors

To solve the error:

1. In the **Validation Results** window select the error and with right click invoke its short-cut menu.
2. In the Containment Tree select the element with error (marked with red circle and white cross inside) and with right-click invoke its shortcut menu.
3. On the diagram pane, select the element with error (highlighted with red border) and with right-click invoke its shortcut menu.

In the shortcut menu of element with error, select the **Correct** command to invoke the **Spell Checker** dialog box.

In the **Spell Checker** dialog box the **Element** property shows the name of the element. The **Property** shows the name of the element's property that has spelling error in its value. In the **Value** property all wrongly spelled words are underlined.

Press the **Next** button to go to the next spelling error found during validation. To close spelling dialog and save changes press **OK**. To close dialog without saving changes press **Cancel**.



Figure 243 -- Solving the spell checking errors in the Validation Results window

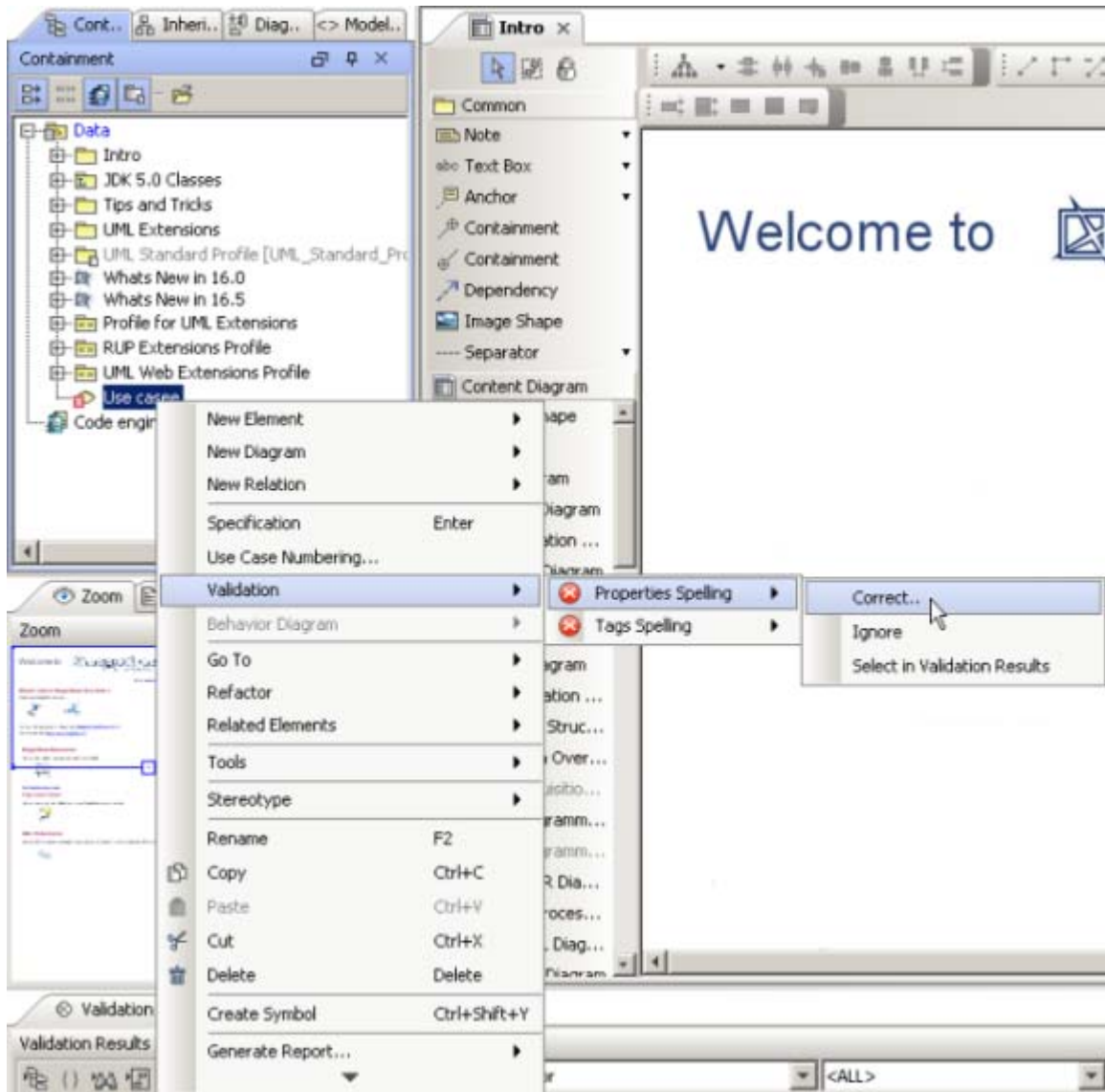


Figure 244 -- Solving spell checking errors in the Containment tree

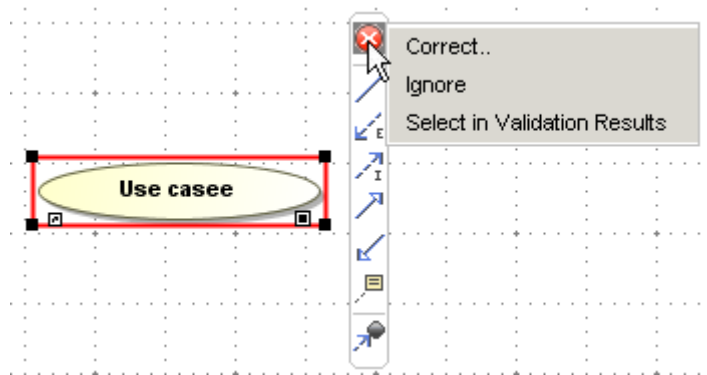


Figure 245 -- Solving spell checking errors in the diagram pane

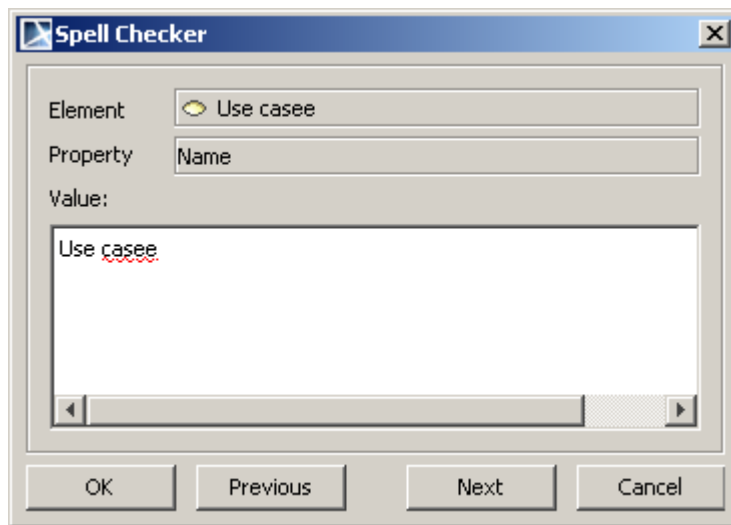


Figure 246 -- The Spell Checker dialog box

Setting the spell checking options

1. From the **Options** main menu, select **Environment**. The **Environment Options** dialog box opens.
2. Select the **Spelling** branch. Define the spelling options.

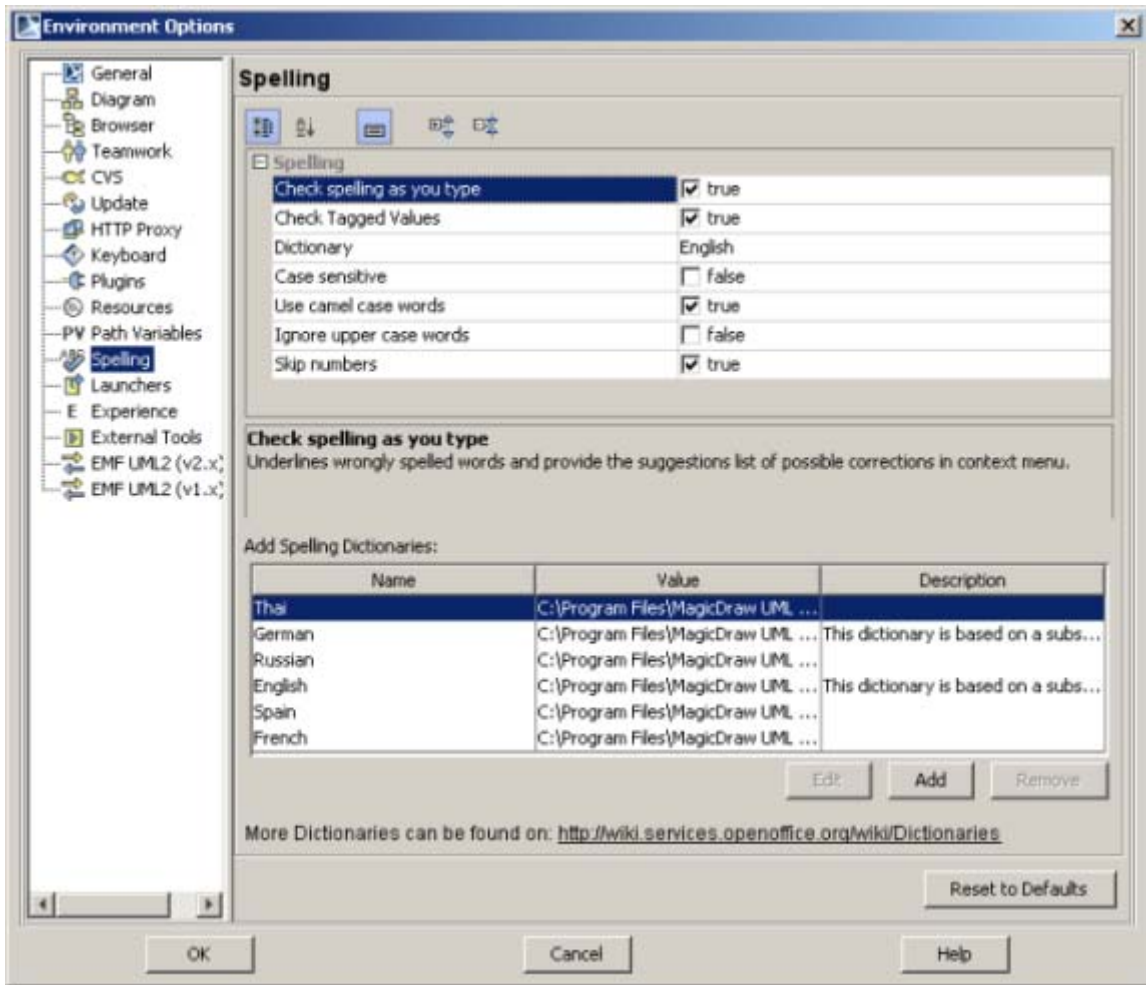


Figure 247 -- The Environment Options dialog box, the Spelling branch

Spell checking options

See the spelling options in the table below.

Property name	Function
---------------	----------

Check Spelling as you type	Underlines wrongly spelled words and provide the suggestions list of possible corrections in context menu. For more information about spell check on type, see "Spell checking as you type" on page 501.
Check Tagged Values	Checks all tagged values that is Type of string.
Dictionary	Select language for spelling. All "Open Office" supportive spelling languages can be added additionally to the existing ones. For more information about spelling dictionaries, see "Spell checking dictionaries" on page 513.
Case Sensitive	If true, words will differ in meaning based on differing use of uppercase and lowercase letters.
Use camel case words	If true, compound words or phrases in which the words are joined without spaces and are capitalized within the compound-as in BackColor, will be spelled as separate words.
Ignore upper case words	If true, words with uppercase words only are not to be spell checked.
Skip numbers	If true, numbers are not to be spell checked.

Spell checking dictionaries

All "Open office" supportive spelling languages are available.

To import spelling dictionaries:

1. Click the **Add** button in the **Environment Options** dialog box > the **Spelling** branch > the **Add Spelling Dictionaries** group. The **Dictionary** dialog box opens.
2. Type the name of a new spelling dictionary in the **Name** text box.
3. Click the "..." button and select the OpenOffice zip file location.

4. Type the description of a new spelling dictionary in the **Description** text box.

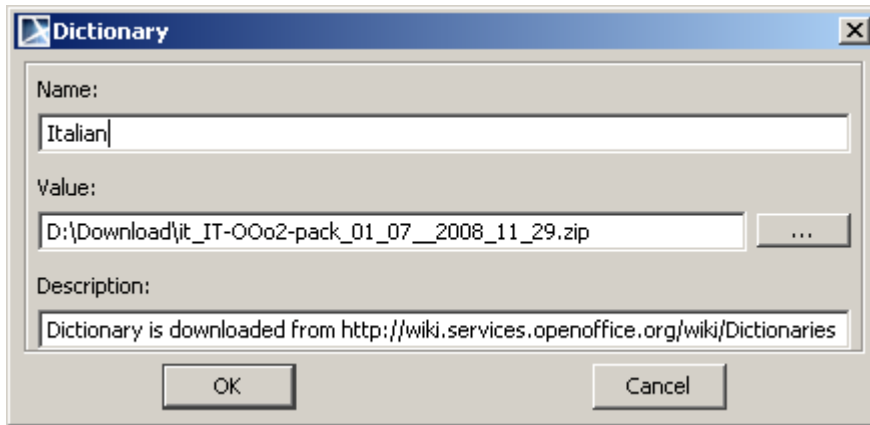


Figure 248 -- The Dictionary dialog box

NOTE More dictionaries can be found on: <http://wiki.services.openoffice.org/wiki/Dictionaryes>

Defining properties of the customized element to be spell checked

DSL customization classes and their properties can be checked either. You can choose what properties of the customized element (Class with <<Customization>> stereotype) you want to be spelling checked.

Defining customization class tag "checkSpelling" value can do it. "checkSpelling" tag can be found in properties tag group. By creating value for this tag you can choose String properties to check spelling for. By default there are no properties marked as checkable.

For more information about DSL, see UML Profiling and DSL UserGuide.pdf.

Import Data to MagicDraw

Import data from Rational Software Architect/Modeler using MagicDraw RSXConverter

MagicDraw RSXConverter provides a seamless way to convert IBM® Rational® Software Architect (RSA) or IBM® Rational® Software Modeler (RSM) file format (*.emx, epx and efx) to MagicDraw-supported file format (*.mdxml).

For more information about MagicDraw RSXConverter see:

MagicDraw RSXConverter

Import data from Rational Rose using MagicDraw RConverter

MagicDraw RConverter provides a seamless way to convert Rational Rose Model file format (*.mdl) to MagicDraw-supported file format (*.xmi).

For more information about MagicDraw RConverter, see:

MagicDraw RConverter plugin

Import data from other tools

- MagicDraw can import most of the model data from the latest Enterprise Architect version. Enterprise Architect does not export 100% standard UML 2.1.1 XMI and this causes some data loss on import. Diagramming information is not imported.
- XDE model import is not available.
- Rational Software Architect/Modeler 6.x model files (not diagrams) can be imported with EMF UML2 1.x import.
- Rational Software Architect/Modeler 7.x model files (not diagrams) can be imported with MagicDraw RSXConverter (see “Import data from Rational Software Architect/Modeler using MagicDraw RSXConverter” on page 515) or EMF UML2 2.x import could be used.

- Rational Rose model files can be imported with MagicDraw RConverter (see “Import data from Rational Rose using MagicDraw RConverter” on page 515).
- Together 2006 model files (not diagrams) can be imported with EMF UML2 1.x import.
- Visio model files can be imported using a XML exporter. Most of the static structure elements (not diagrams) can be imported. XML plugin for Visio can be found:

Executable file of the XML plugin	Visio versions	Link
XMIExport.exe	Microsoft Visio 2002 Professional, Microsoft Visual Studio .NET Enterprise Architect (Visio for Enterprise Architects)	http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=BE6D20EF-36BA-4ABF-A26F-91434C7E7B7
XMIExprt.exe	Microsoft Visio 2003	http://www.microsoft.com/downloads/details.aspx?familyid=3DD3F3BE-656D-4830-A868-D0044406F57D&displaylang=en#Overview

8 MODEL ANALYSIS

MagicDraw provides the following analysis capabilities to help you analyze your model:

- **"Model Visualizer"**:
 - "Class Diagram Wizard" – helps to create and customize new class diagrams.
 - "Package Overview Diagram Wizard" – generates the package dependency diagram for packages in your project.
 - "Package Dependency Wizard" - generates diagrams containing packages (created within a project) and shows the relationships between them.
 - "Hierarchy Diagram Wizard" and "Realization Diagram Wizard" – prepares diagrams and report documents of the relationships between classes in the UML model.
 - "Activity Decomposition Hierarchy Wizard" - converts activity into class and/or SysML Block Definition Diagram.
 - "Content Diagram Wizard" - generates content of diagrams that are used in the project.
 - "Sequence Diagram from Java Source Wizard" - creates a sequence diagram of Java method implementation.
- **"Displaying related elements"** - displays paths among shapes that have already been created in the model data, use the quick and simple Display related elements functionality.
- **"Dependencies analysis"** - the MagicDraw Usages and Dependencies feature enables you to track and view element dependencies in UML models, explore how model elements are used by other elements, and understand the relationships between used and dependent elements.
- **"Model Differencing"** - compares two MagicDraw UML local projects or teamwork project versions, as well as diagrams.
- **"Metrics"** - allows measuring a project by different viewpoints.
- **"Dependency Matrix"** - is a method of visualizing and representing dependency criteria.

- **"Validation"** - a facility for evaluating completeness and correctness of the models created by the user.
- **"Active Validation"** - instantly checks model for correctness and completeness, displays errors in the model and suggests solutions.
- **"Usage in Diagrams"** - validates if the symbol owner on the diagram matches the actual element owner in the model.
- **"Usage in Diagrams"** - Displaying the list of diagrams in which symbol of current element is represented.

Model Visualizer

NOTE

This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

MagicDraw contains tools that help to create elements from existing data and analyze the relationships between elements created in the UML model. It is also possible to analyze the inheritance and dependency relationships between classes.

All model visualizing and analyzing tools are presented in MagicDraw as Wizards with several steps that should be followed in order to accomplish the desired operation.

All Wizards have several common buttons:

- **Add** – add selected model elements from the **All** list to the **Selected** list.
- **Add All** – add all elements to the **Selected** list that are located in the same hierarchy level as the selected element.
- **Add Recursively** – add all elements in the selected packages and all elements from nested packages to the **Selected** list.
- **Remove** – remove the selected element from the **Selected** list.
- **Remove All** – remove all selected elements.
- **< Back** – return to the previous dialog box.
- **Next >** – proceed to the next step.

- **Finish** – finish the configuration. All other options will be set by default. The **Wizard** exits and results are displayed.
- **Cancel** – cancel the wizard.
- **Reset To Defaults** - if changes were made to the element properties, values will be set to default.

Class Diagram Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Class Diagram Wizard** helps you create a new class diagram when all classes and their relationships are already created and specified. You can select which classes, packages, and relationships will be included in a new class diagram and the details of the class representation to be configured (attributes, operations, accessibility). The **Class Diagram Wizard** frees you from creating the class diagram manually. The **Class Diagram Wizard** guides you through several steps and collects information along the way. It will automatically create a new class diagram and all the necessary elements.

To start the **Class Diagram Wizard**

1. From the **Analyze** menu, select **Model Visualizer**. The **Model Visualizer** dialog box opens.
2. From the wizards list, select the **Class Diagram Wizard**.

3. Click **Start**. The **Class Diagram Wizard** opens.

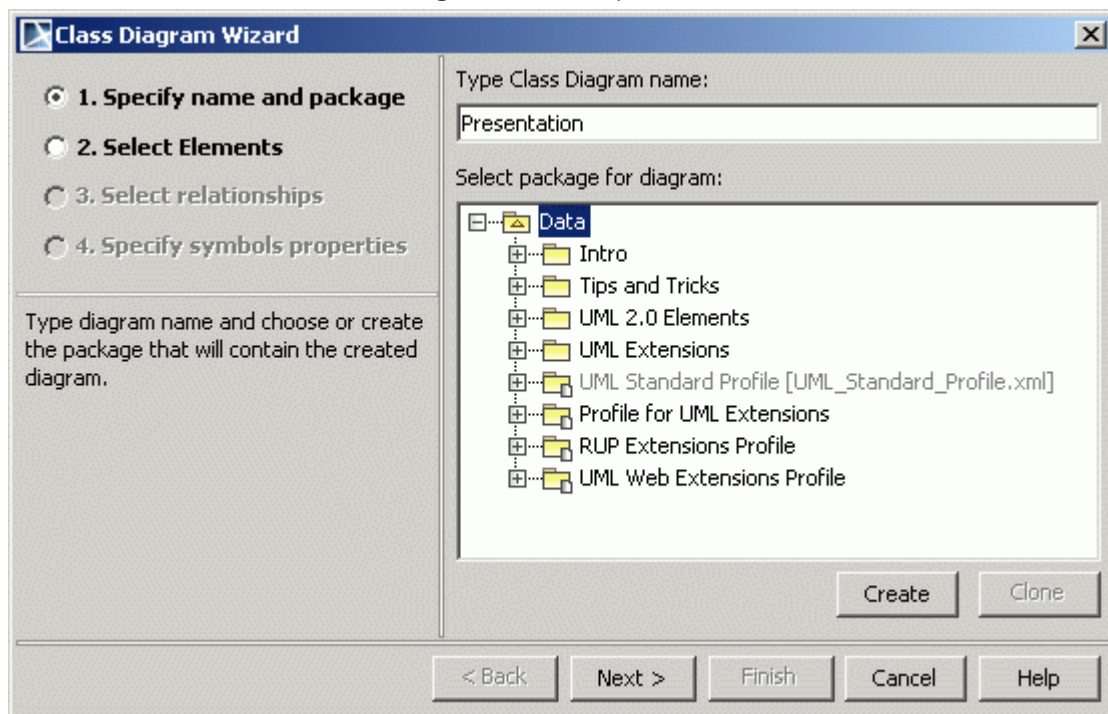


Figure 249 -- Class Diagram Wizard. Diagram name

Type the name of the new diagram in the **Type Class Diagram name** text box or leave the default name.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package from the data tree that will be the parent for the newly created diagram or create a new package by clicking the **New** button.

When you select the **Specify name and package** option, the following functions are available:

- **Create** - create a new package.

- **Clone** - copy an existing package to a new one.

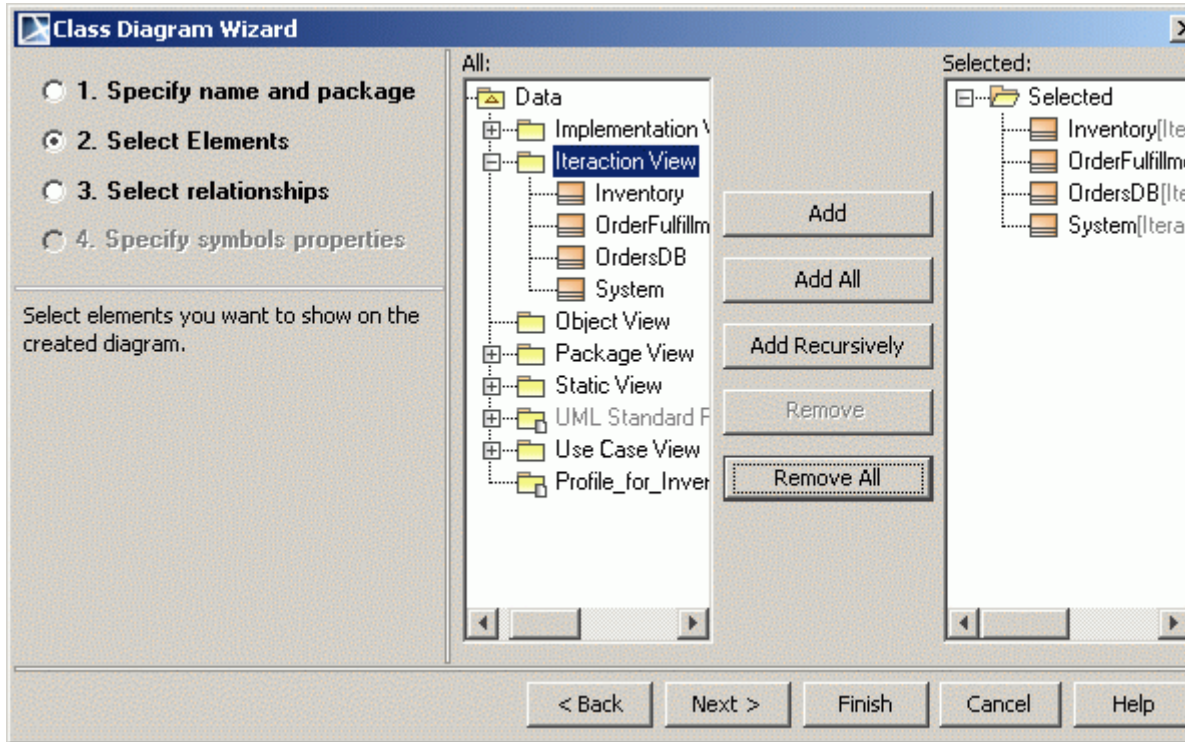


Figure 250 -- Class Diagram Wizard. Select Elements

- **All list** – contains all model elements.

- **Selected** list – contains the elements that are added to the class diagram.

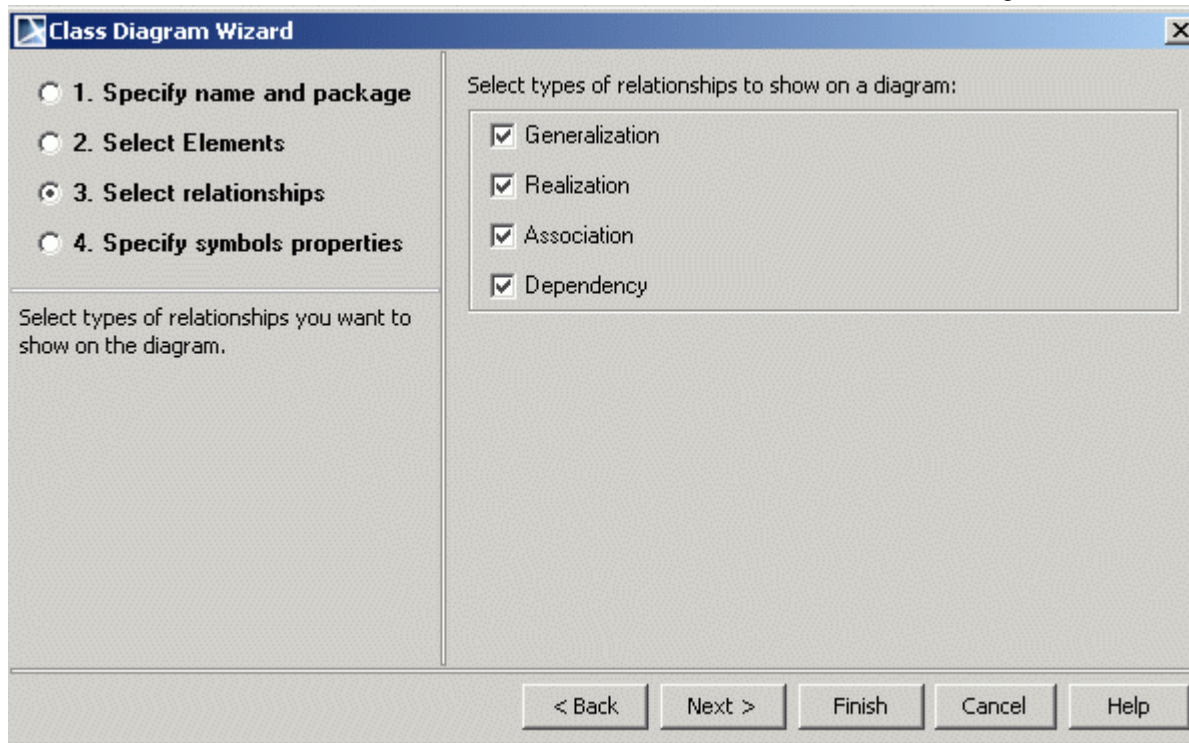


Figure 251 -- Class Diagram Wizard. Select Relationships

Select relationships to include in the class diagram:

- **Generalization** – relationship between a general element and a more specific element (inheritance, extension).
- **Realization** – relationship between model elements where one of the elements implements the interface defined by the other model element.
- **Association** – semantic relationship between classes.

- **Dependency** – usage relationship between UML model elements.

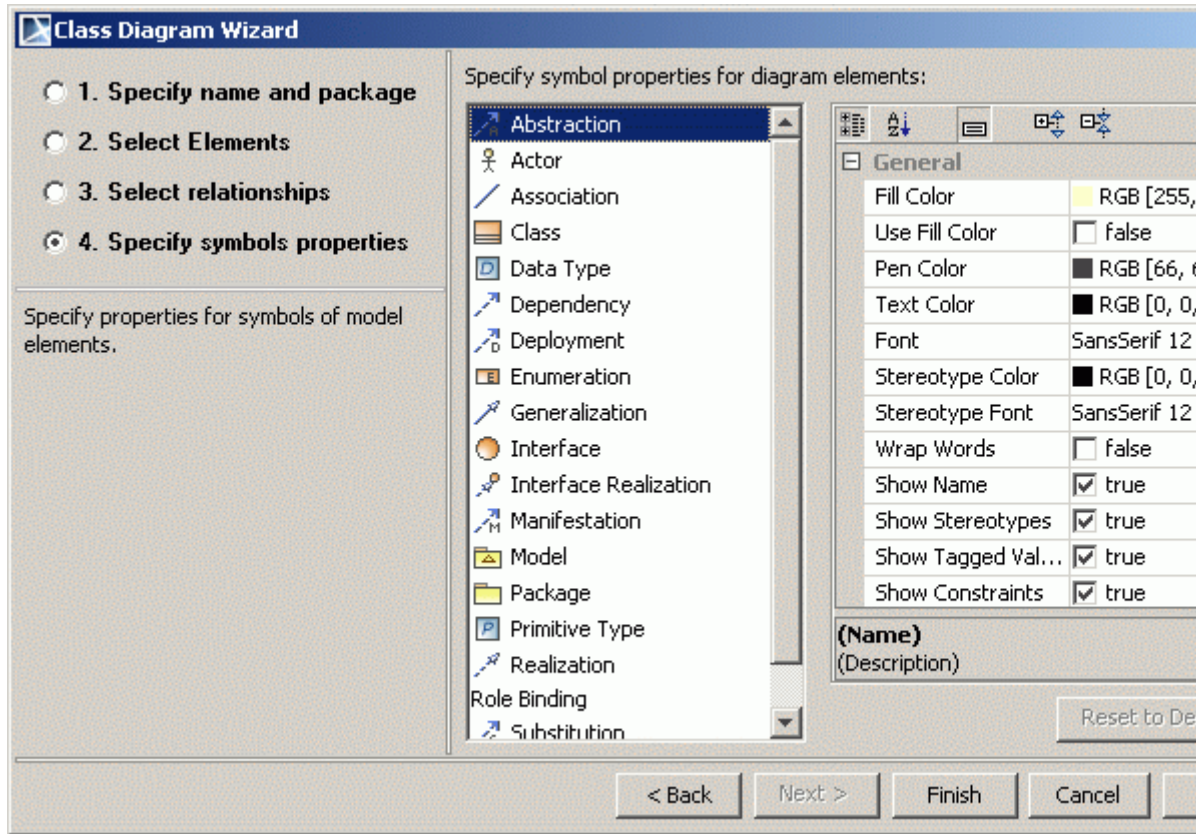


Figure 252 -- Class Diagram Wizard. Specify Symbols Properties

Select options for representing elements in the class diagram.

NOTE

If the **Suppress Attributes** and the **Suppress Operations** check boxes are selected, the class is displayed only as a rectangle with the class name in it.

Package Dependency Wizard

NOTE

This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Package Dependency Diagram Wizard** generates diagrams containing packages (created within a project) and shows the relationships between them. The diagram may reflect all packages in the project, or just those selected. The **Package Dependency Diagram Wizard** collects all the information needed to both analyze dependencies and generate a new diagram.

To start the **Package Dependency Diagram Wizard**

1. From the **Analyze** menu, select **Model Visualizer**. The **Model Visualizer** dialog box opens.
2. From the wizards list, select the **Package Dependency Diagram Wizard**.

3. Click **Start**. The **Package Dependency Diagram Wizard** opens.

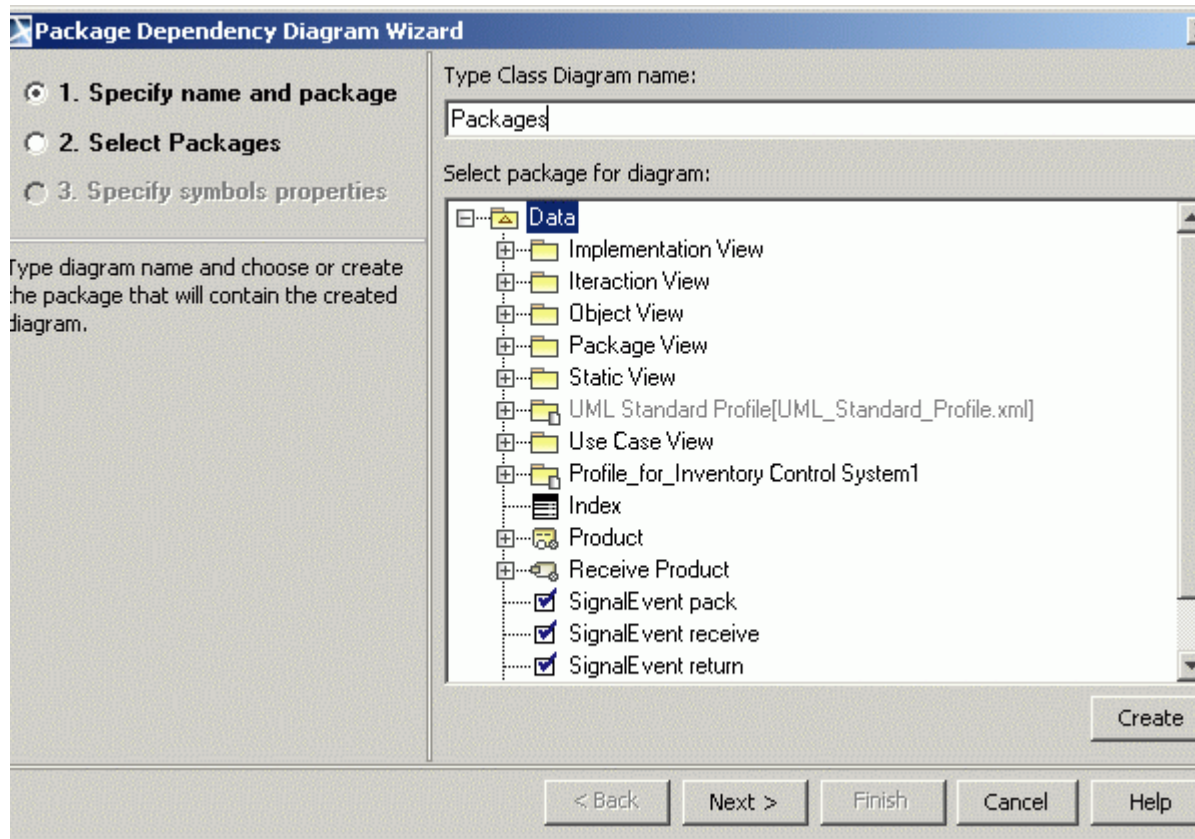


Figure 253 -- Package Dependency Diagram Wizard. Diagram name

Type a name for the newly created diagram in the **Type Class Diagram name** text box.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package from the data tree that

will be the parent for the newly created diagram or create a new package by clicking the **Create** button.

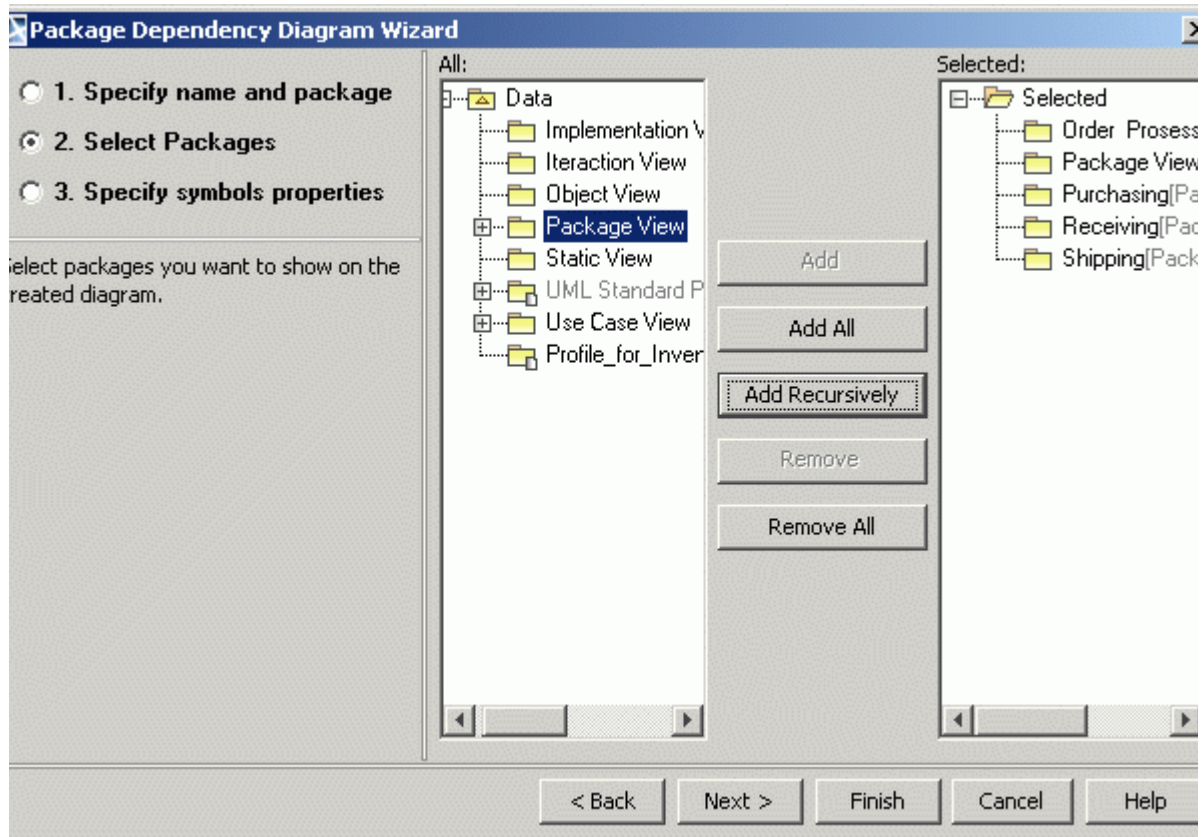


Figure 254 -- Package Dependency Diagram Wizard. Select Package.

- **All** list – contains all packages.

- **Selected** list – contains all packages that are added to the diagram.

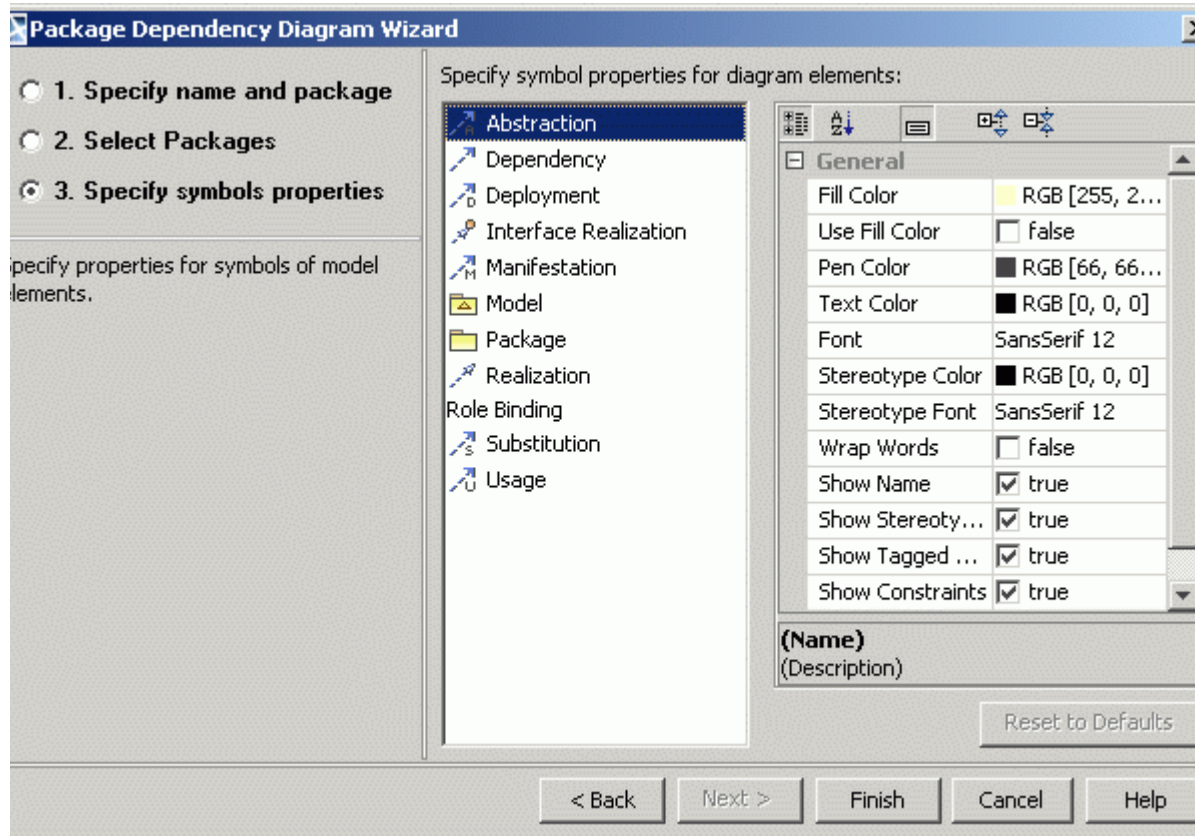


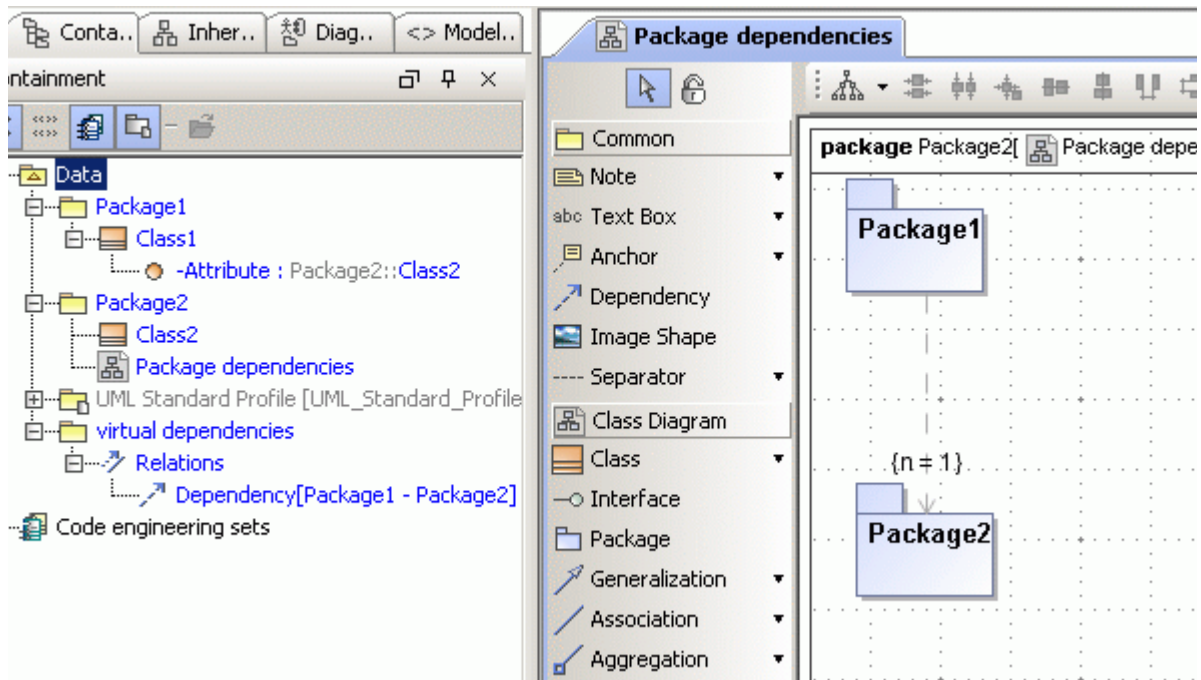
Figure 255 -- Package Dependency Diagram Wizard. Specify Symbols Properties

Select options for representing elements in the class diagram.

Displaying package dependencies

If a package has inner elements used by or dependent on other package elements, the Package Dependencies Wizard will analyze dependencies and create “virtual” relations between dependent packages.

For example:



The **Virtual dependencies** package containing dependency links is created after finishing the wizard. If there is more than one dependency between the package inner elements, then the tagged value number {n=...} on the virtual dependency will be changed according to the number of dependencies.

Package Overview Diagram Wizard

NOTE

This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Package Overview Diagram Wizard** allows the creation of a diagram for every package from the selected scope (reversed packages). The created diagram displays the content of the packages – inner packages with inner elements connected with available relations.

To start the **Package Overview Diagram Wizard**

- Select the **Package Overview Diagram Wizard** from the **Diagrams** menu, **Diagram Wizards** submenu.
- Open the package shortcut menu, select **Tools** and then **Package Overview Diagram Wizard**.
- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. In the wizards list, select the **Package Overview Diagram Wizard**. Click the **Start** button.

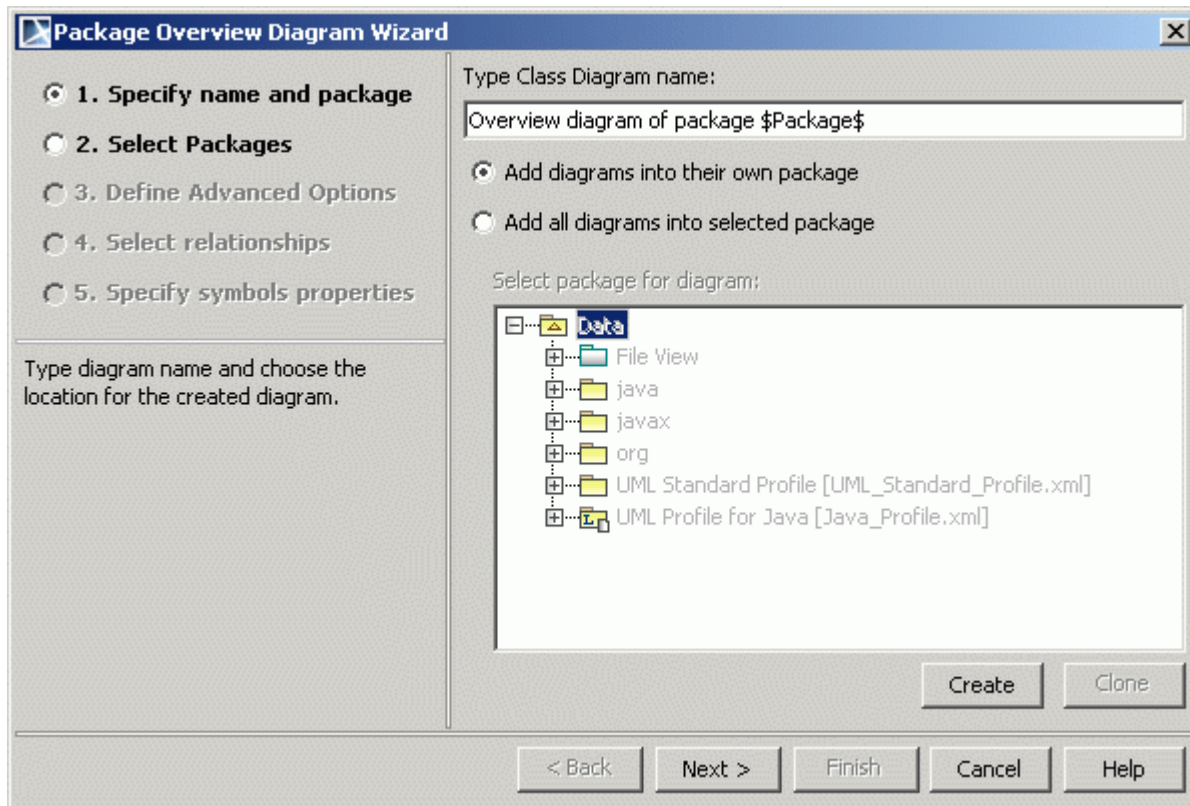


Figure 256 -- Package Dependency Diagram Wizard. Diagram name

Type a name for the newly created diagram in the **Type Class Diagram name** text box.

The **Add diagrams into their own package** option button - adds diagrams in the same package they are created.

The **Add all diagrams into selected package** - while creating new package overview diagrams, adds diagrams in the selected package. Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package that will be the parent for the newly created diagram from the Data tree or create a new package by clicking the **New** button.

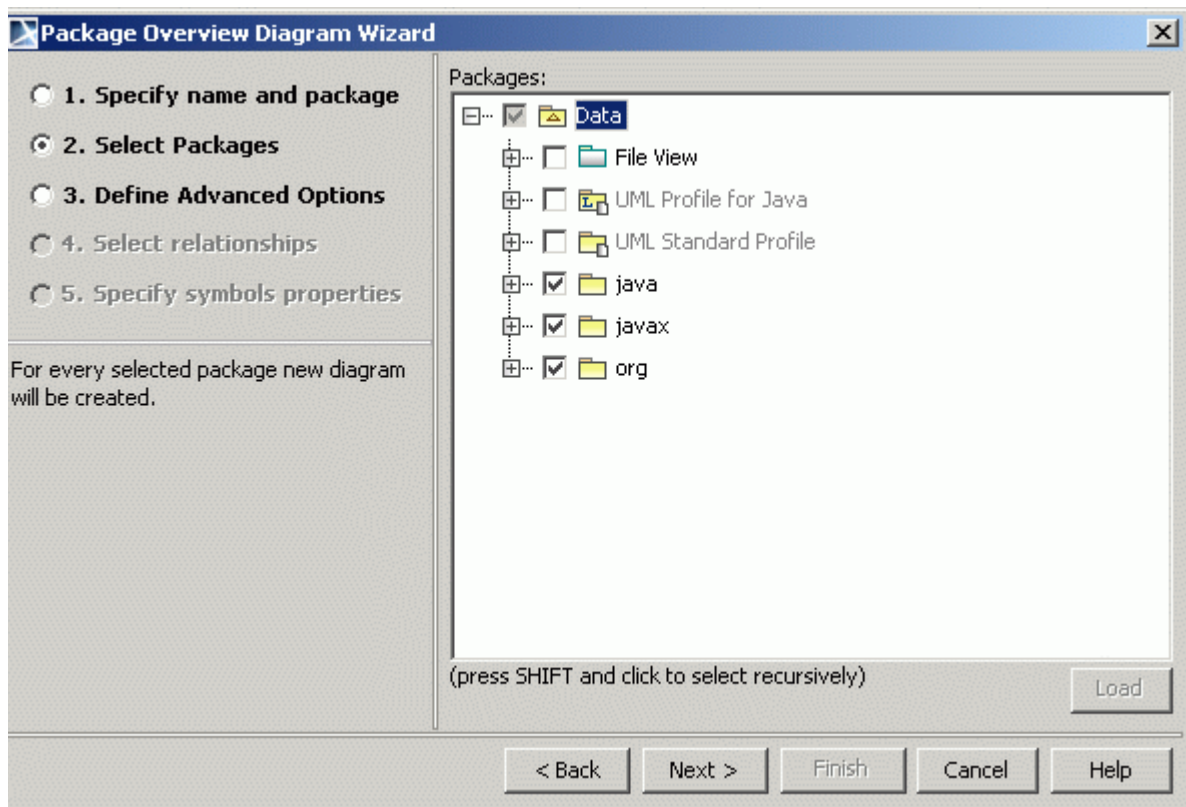


Figure 257 -- Package Overview Diagram Wizard. Select Package.

Select the packages, which will be represented in the new diagram. For every selection, a package diagram will be created.

If the selected package is read-only and the package for the diagrams is not specified, a warning will be displayed when the **Next** button is pressed.

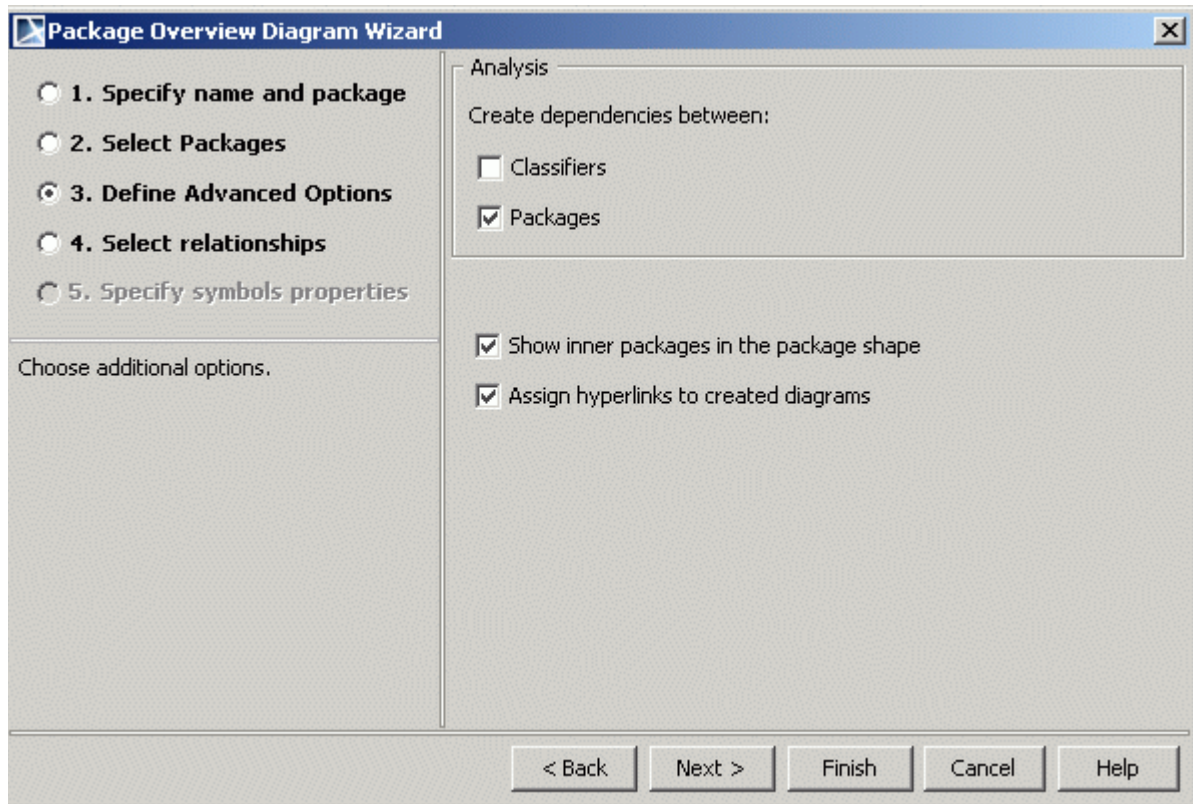


Figure 258 -- Package Overview Diagram Wizard. Define Advanced Options

Set the advanced properties for elements to be represented in the diagrams.

If you want to see the classifiers structure in the created class diagram, then select the create dependencies between **Classifiers** check box in the Analysis options group.

If create dependencies between **Packages** check box is selected, then only the package content class diagram will be created. Analyzes are performed of all inner elements, recursively by all criteria.

The **Show inner packages in the package shape** check box - displays one level of inner packages in every package shape, connected with dependencies.

The **Assign hyperlinks to created diagrams** check box - adds an active hyperlink to every package, referenced to the inner diagram of this package.

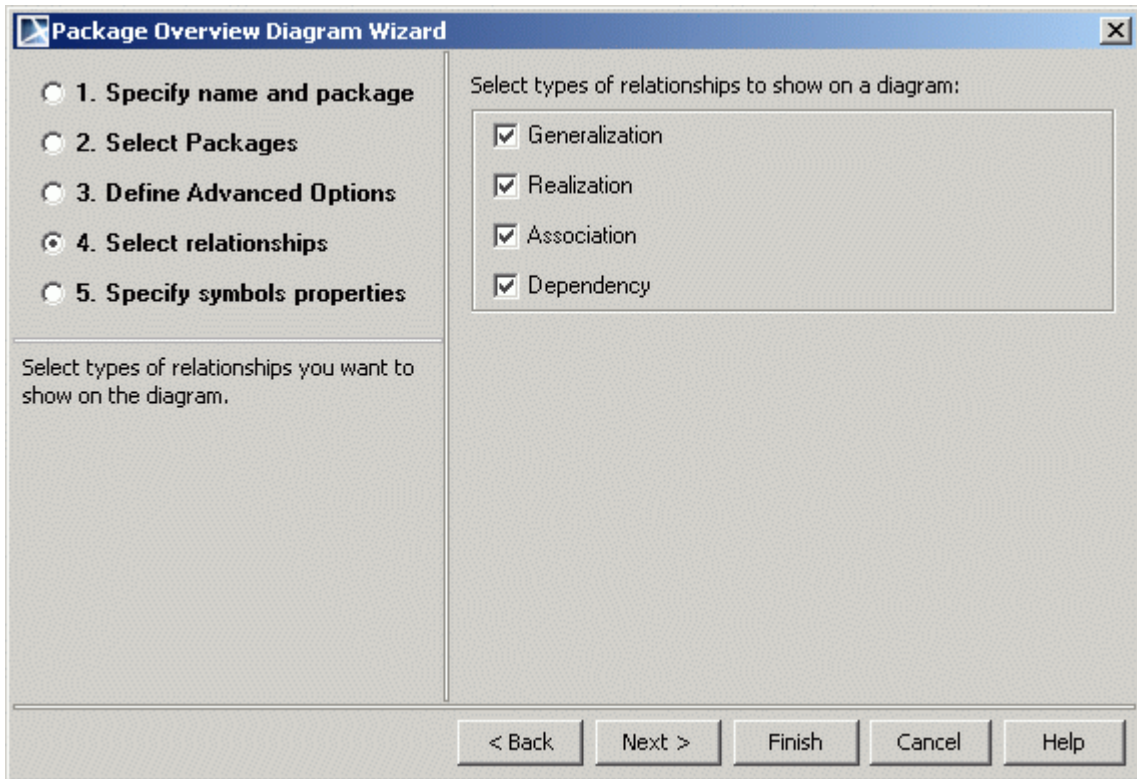


Figure 259 -- Package Overview Diagram Wizard. Select Relationships

Select the relationships you wish to include in the class diagram:

- **Generalization** – relationship between a general element and a more specific element (inheritance, extension).
- **Realization** – relationship between model elements where one of the elements implements the interface defined by the other model element.
- **Association** – semantic relationship between classes.

- **Dependency** – usage relationship between UML model elements.

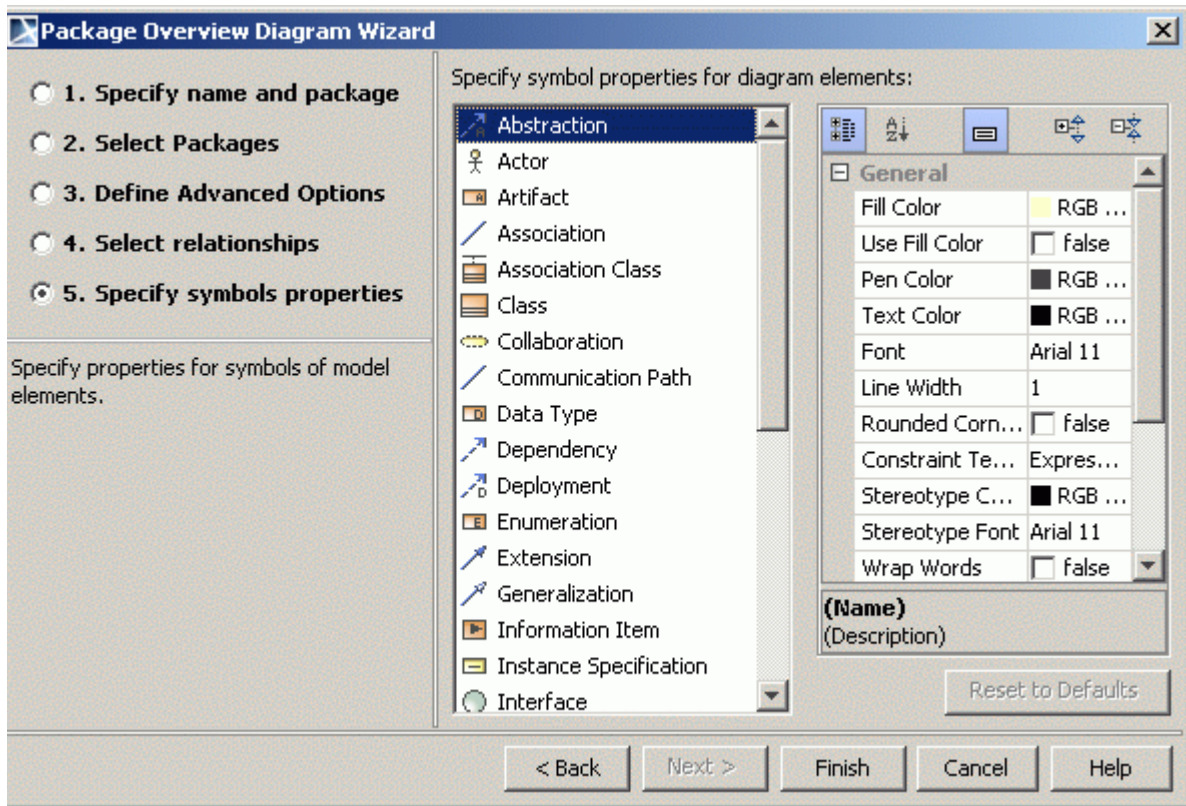


Figure 260 -- Package Overview Diagram Wizard. Specify symbols properties

Select options for representing elements in the diagram.

Hierarchy Diagram Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Hierarchy Diagram Wizard** collects the largest hierarchies and allows every of them to be displayed as separate diagrams or all in one diagram.

To start the **Hierarchy Diagram Wizard**

- From the **Diagrams** main menu, select the **Diagram Wizards** command and then **Hierarchy Diagram Wizard**.
- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. From the wizards list, select the **Hierarchy Diagram Wizard**.
- From the model element shortcut menu, select **Tools** and then **Hierarchy Diagram Wizard**.

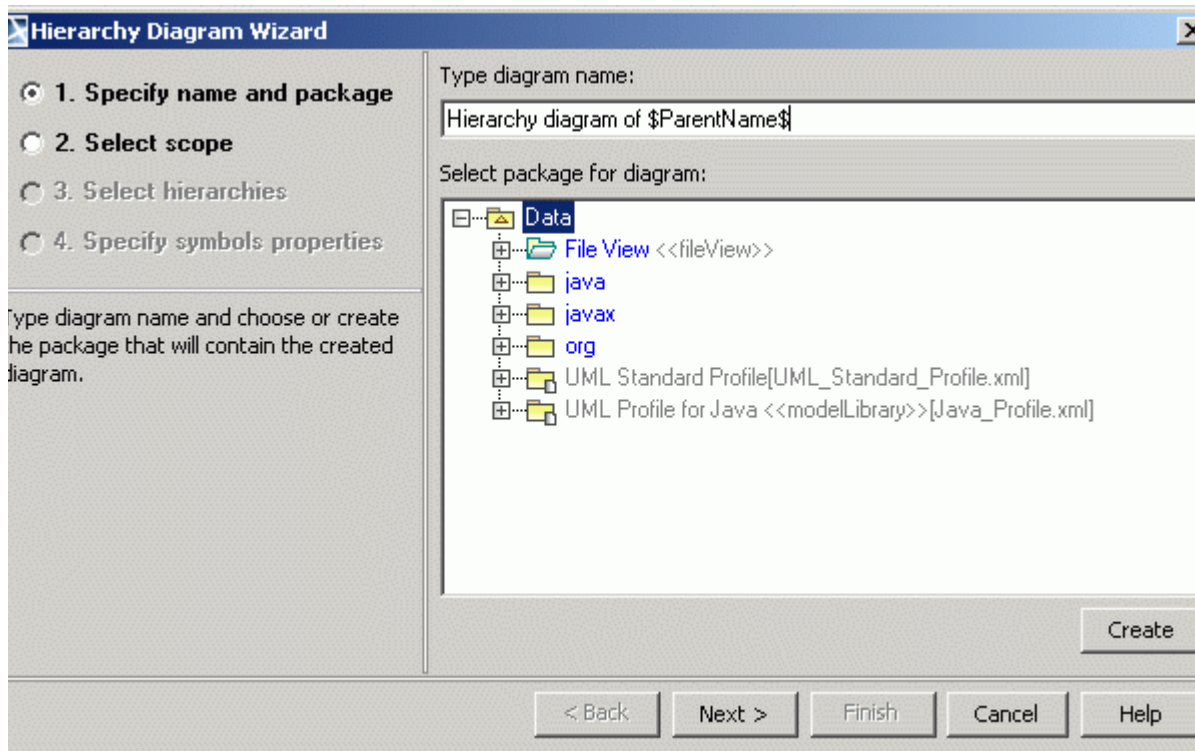


Figure 261 -- Hierarchy Diagram Wizard. Specify Name and Package

Type a name for the new diagram in the **Type diagram name** field.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package that will be the parent

for the newly created diagram from the Data tree or create a new package by clicking the **Create** button.

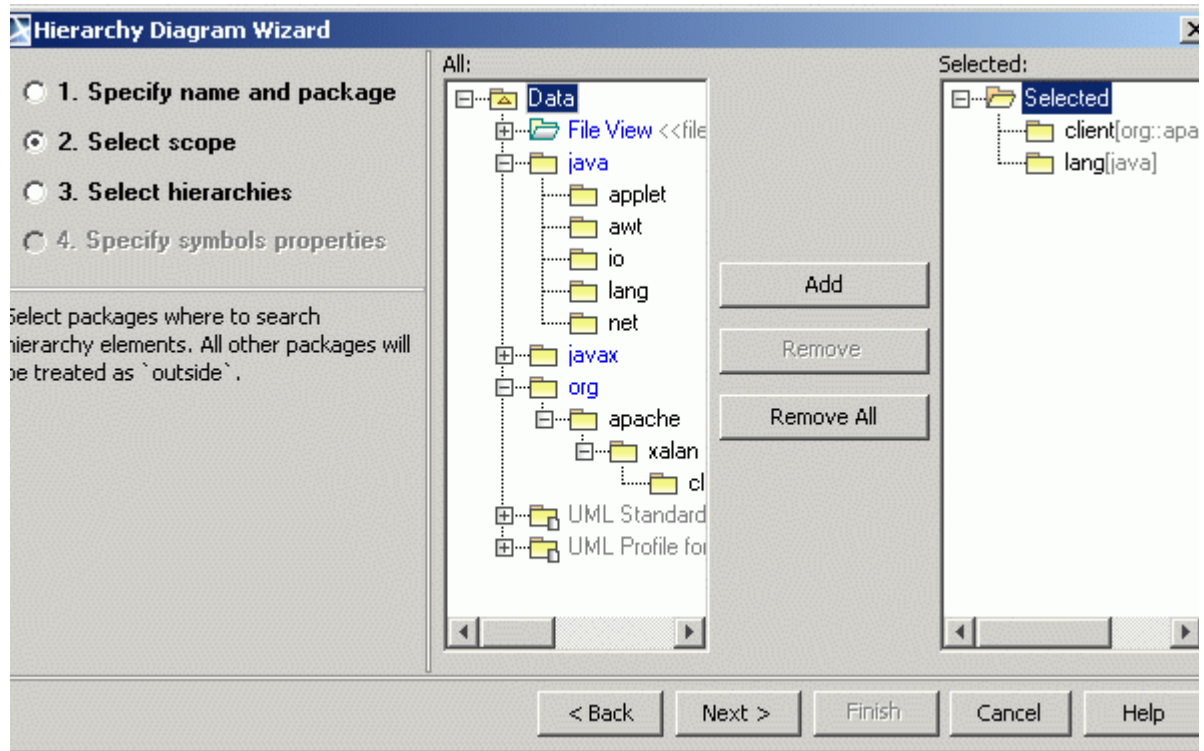


Figure 262 -- Hierarchy Diagram Wizard. Select Scope

Select packages from the **All** list to the **Selected** list.

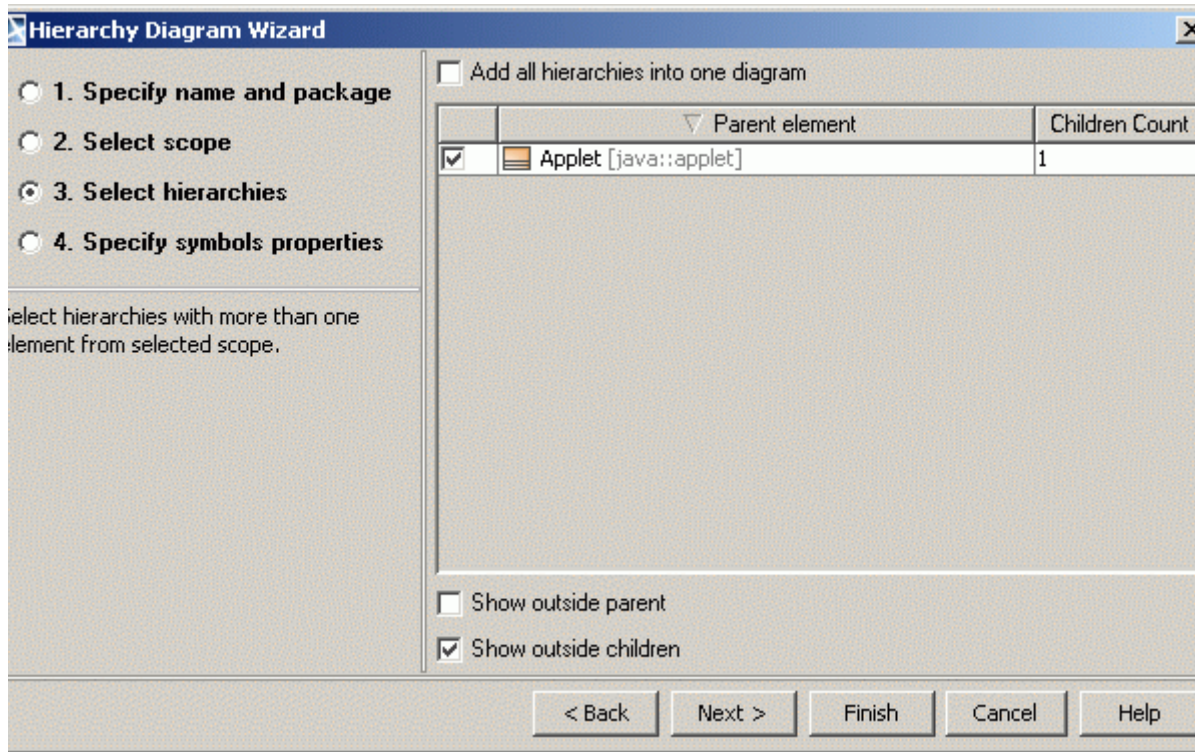


Figure 263 -- Hierarchy Diagram Wizard. Select Hierarchies.

The **Add all hierarchies into one diagram** check box creates only one diagram for all selected hierarchies. This option is enabled only if the selected hierarchies can be added into one diagram (the same diagram type).

All available hierarchies are listed in the **Parent Element** column. In the **Children Count** column, the number of model elements is presented.

The **Show outside parent** check box shows hierarchies, when derived packages are in the scope, but specializations is from outside the scope.

The **Show outside children** check box counts outside derived elements from the displayed hierarchies. Otherwise the hierarchy will not be fully displayed and the diagram may be not valid.

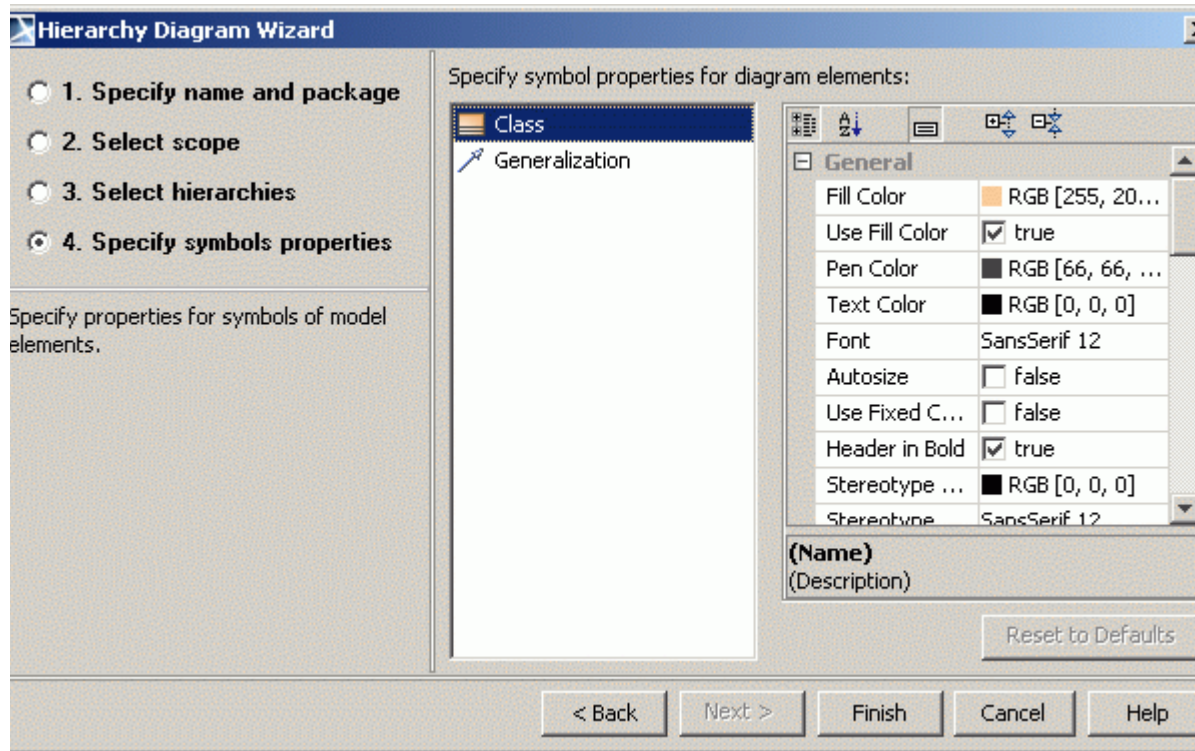


Figure 264 -- Hierarchy Diagram Wizard. Specify Symbols Properties

Select options for representing elements in the diagram.

Realization Diagram Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Realization Diagram Wizard** shows a table of the largest element groups that realize some interface.

To start the **Realization Diagram Wizard**

- From the **Diagrams** main menu, select the **Diagram Wizards** command and then **Realization Diagram Wizard**.
- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. From the wizards list, select the **Realization Diagram Wizard**.
- From the model element shortcut menu, select **Tools** and then **Realization Diagram Wizard**.

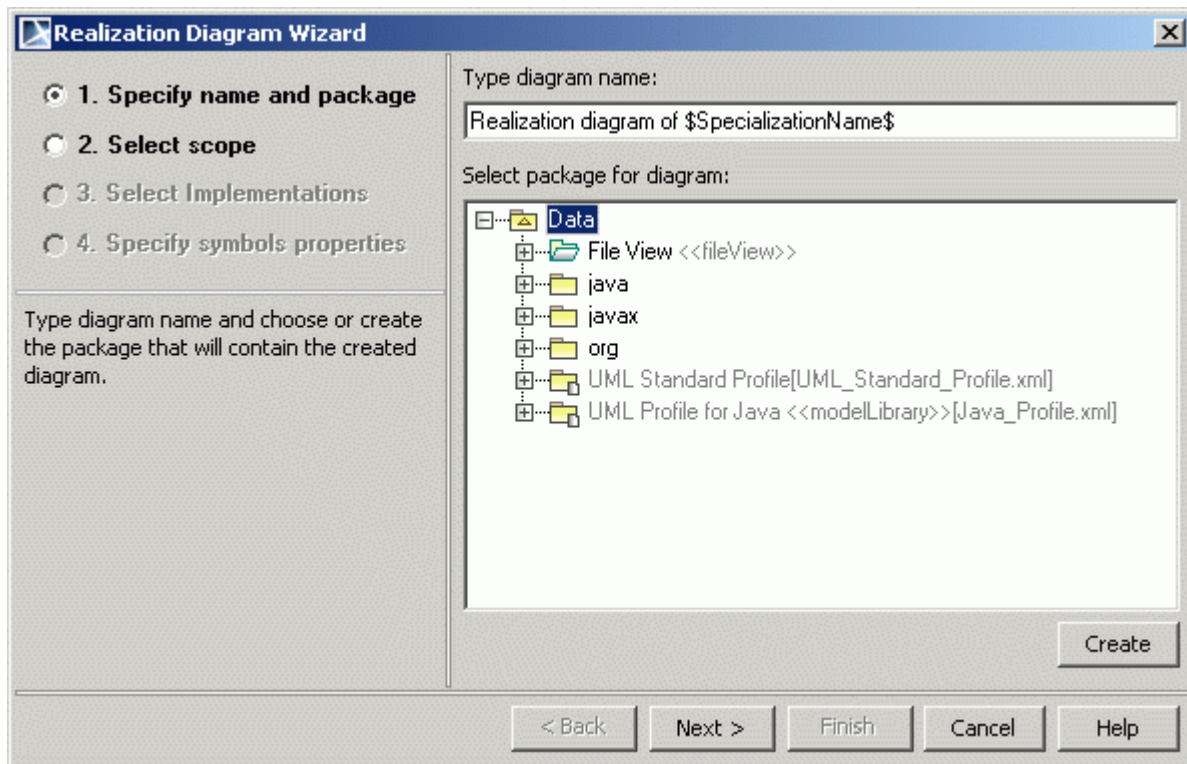


Figure 265 -- Realization Diagram Wizard. Specify Name and Package

Type a name for the new diagram in the **Type diagram name** field.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package that will be the parent

for the newly created diagram from the Data tree or create a new package by clicking the **Create** button.

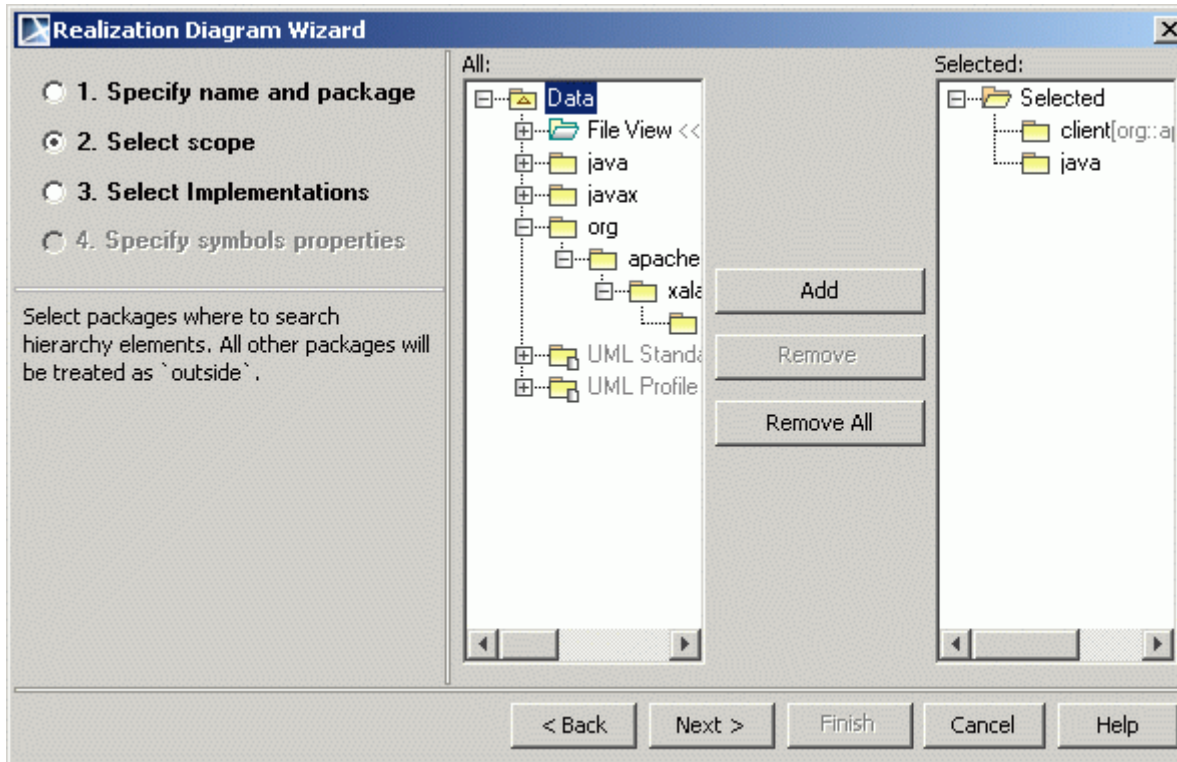


Figure 266 -- Realization Diagram Wizard. Select Scope

Select packages from the **All** list and add them to the **Selected** list to search for hierarchy elements.

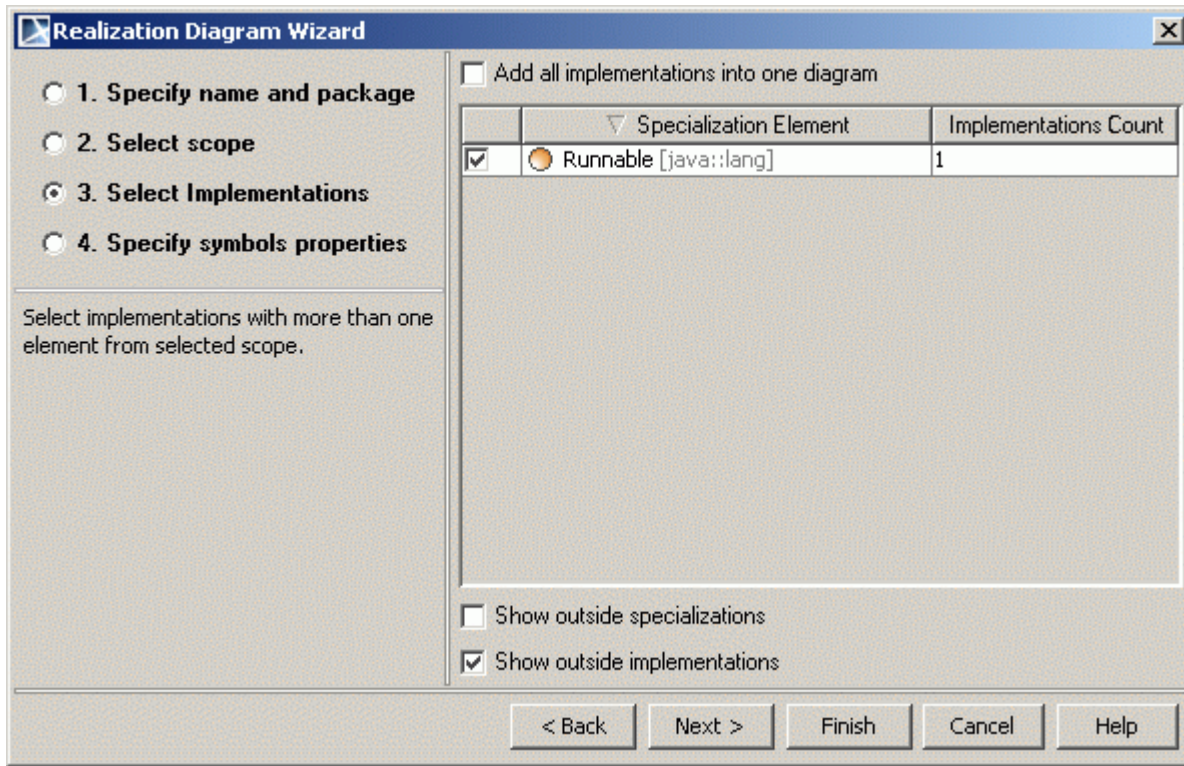


Figure 267 -- Realization Diagram Wizard. Select Implementations

The **Add all implementations into one diagram** check box creates only one diagram for all selected realizations. This option is enabled only if the selected realizations can be added into one diagram (the same diagram type).

All available implementations are listed in the **Specialization Element** column. In the **Implementations Count** column, the number of model elements is presented.

The **Show outside specializations** check box show realizations, when derived interfaces are in the scope, but specializations is from outside the scope.

The **Show outside implementations** check box count outside derived elements from displayed realizations. Otherwise realization will not be fully displayed and the diagram cannot be valid.

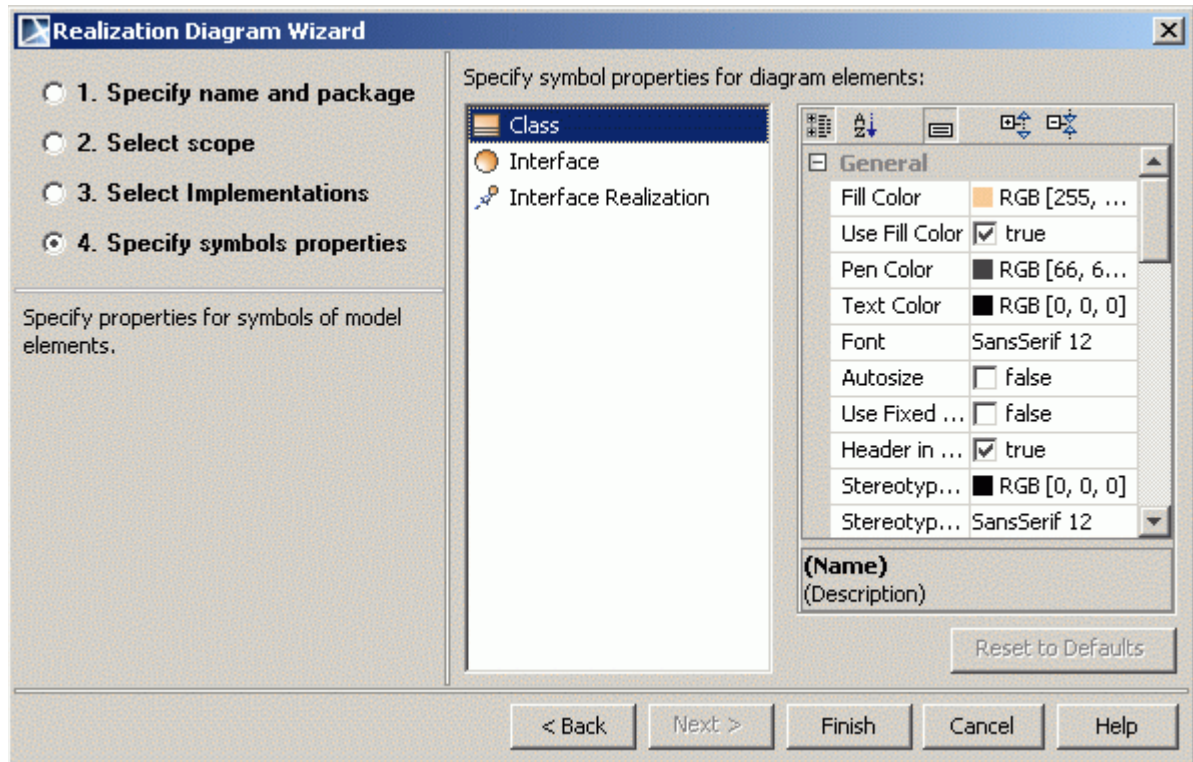


Figure 268 -- Realization Diagram Wizard. Specify Symbols Properties

Select options for representing elements in the diagram.

Activity Decomposition Hierarchy Wizard

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

Activity Decomposition Hierarchy Wizard allows converting activity into class and SysML Block Definition Diagram. This gives the capability to represent, analyze, and document activity hierarchies in the structure diagrams.

Diagram generation rules:

- Behaviors will be connected with contained object node types by compositions. The name of the object node that corresponds to the composition will be used as the end name of the association on the end towards the object node type.
- Pins are not included in the calculation.
- CallBehaviorActions that are not directly in the Activity, but are in the Structured Activity Nodes contained by the Activity, for example, are also included in the calculation.
- Activity will be connected by composition association with other behaviors that are called by CallBehaviorActions . The part end name must be the same as the name of a CallBehaviorAction in the composing activity. If the action has no name, then the end name is as same as that of the invoked activity.
- Hierarchical layout - Top to Bottom is used to arrange the generated diagram.
- If CallBehaviorAction calls the same activity, the composition to self will be displayed on the generated diagram.

Recursive structure analysis will be stopped after reaching the same behavior, which has already been analyzed. This requirement prevents an endless cycle.

 - In such a case, the composition will be created in a previously analyzed activity in the diagram. The new behavior symbol will not be created.
 - There will be as many compositions from one activity into another as different CallBehaviorActions call this activity.

To start the **Activity Decomposition Hierarchy Wizard**

- From the **Diagrams** main menu, select the **Diagram Wizards** command and then **Activity Decomposition Hierarchy Wizard**.

- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. From the wizards list, select the **Activity Decomposition Hierarchy Wizard**.

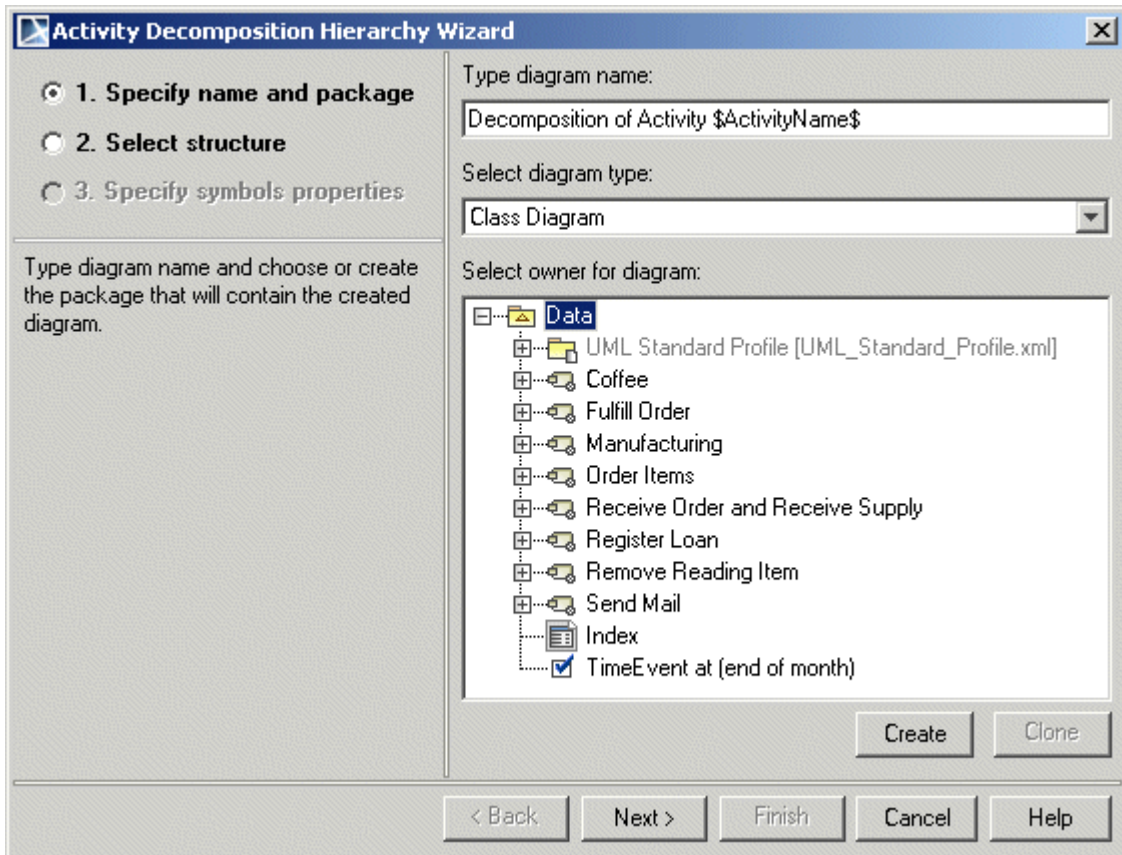


Figure 269 -- Activity Decomposition Hierarchy Wizard. Specify name and package tab

In the **Specify name and package** tab, type the diagram name, select the diagram type the activity will be converted and select or create a package that will contain the desired activity diagram.

NOTE

If you are using the SysML plugin, SysML Block Definition and Class diagrams are available as diagram types. For other domains this list depends on a plug-in of those domains.

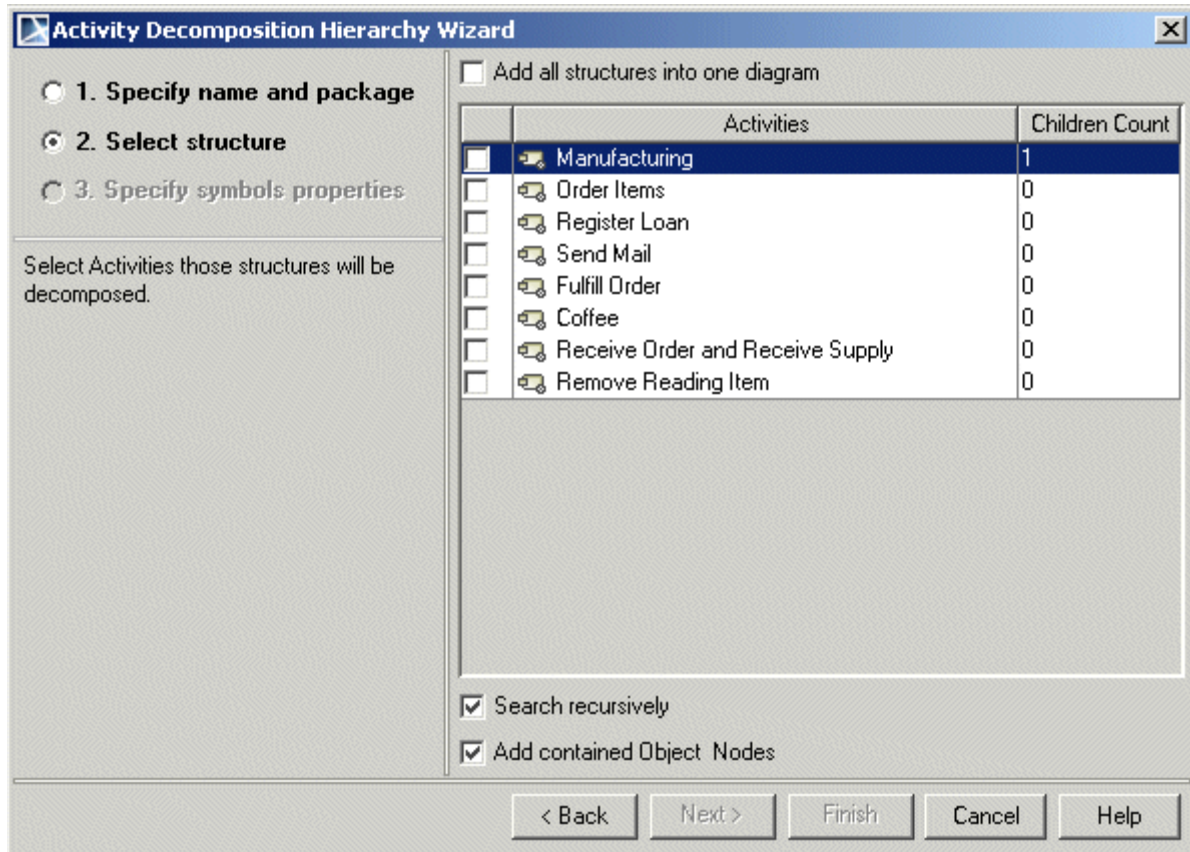


Figure 270 -- Activity Decomposition Hierarchy Wizard. Select structure tab

In the **Select structure** tab, select Activities structures that will be decomposed.

To add all activity structures into one diagram, select **Add all structures into one diagram** check box.

Select check boxes of the desired activity structures. The **Children Count** column shows the number of included behaviors (also owned object nodes if the **Add contained Object Nodes** check box is selected). The number also depends on the option **Search recursively**.

- The **Add contained Object Nodes** check box is selected by default. If selected, types of object nodes are displayed and connected to the composition with activities containing object nodes.
- The **Search recursively check box** is selected by default:
 - If not selected, the search will be conducted in only one level of the selected activity.
 - If selected, the search will be conducted in the selected activity and those activities that are invoked by CallBehaviorActions that are in the selected activity. This search is recursive.

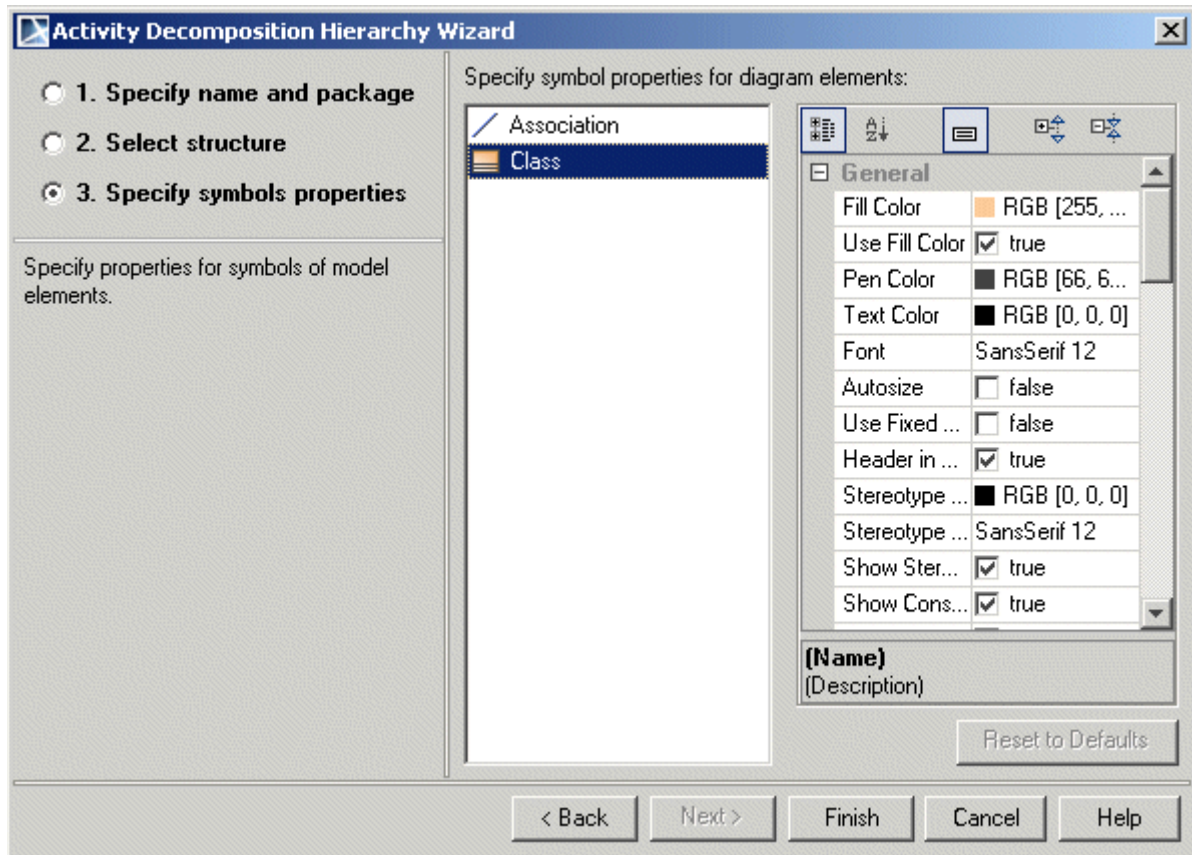


Figure 271 -- Activity Decomposition Hierarchy Wizard. Specify symbols properties tab

Specify properties for symbols for model elements.

Content Diagram Wizard

NOTE

This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The **Content Diagram Wizard** generates content of all diagrams that are usable in the project.

To start the **Content Diagram Wizard**

- From the **Diagrams** main menu, select the **Diagram Wizards** command and then **Content Diagram Wizard**.

- From the **Analyze** menu, select the **Model Visualizer** command. The **Model Visualizer** dialog box opens. From the wizards list, select the **Content Diagram Wizard**.

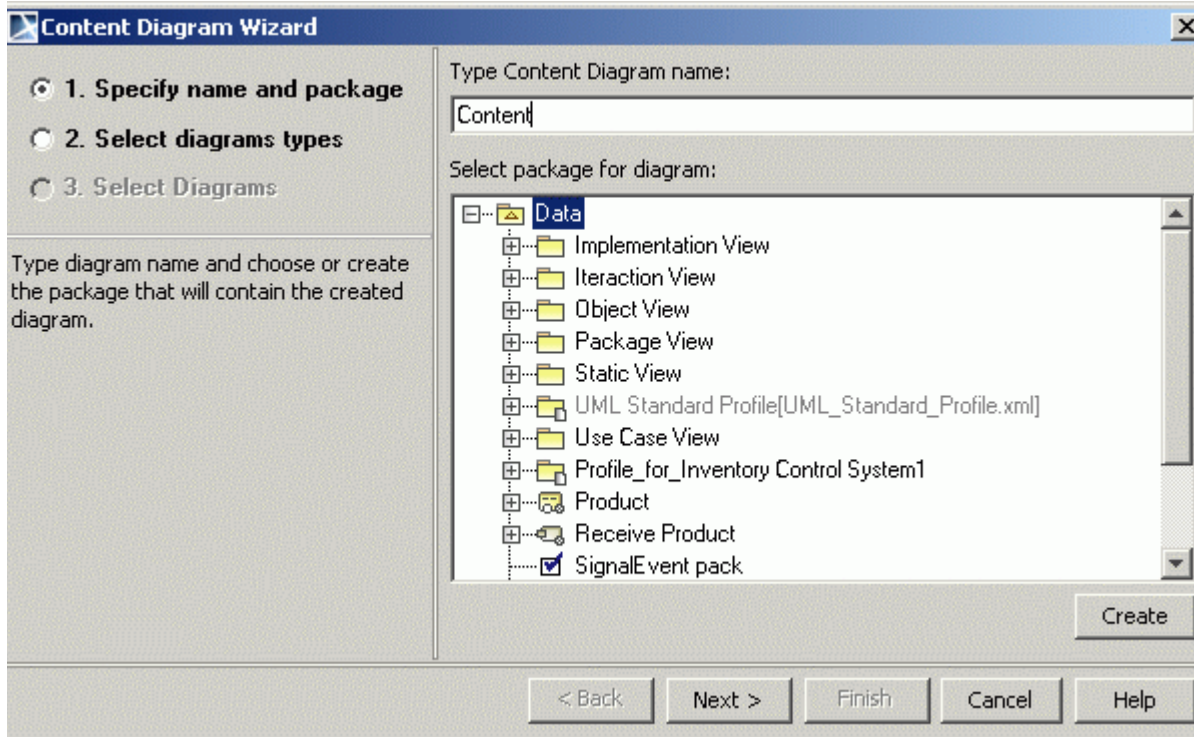


Figure 272 -- Content Diagram Wizard. Specify Name and Package

Type a name for the new diagram in the **Type Content Diagram name** text box.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. Select the package that will be the parent

for the newly created diagram from the Data tree or create a new package by clicking the **Create** button.

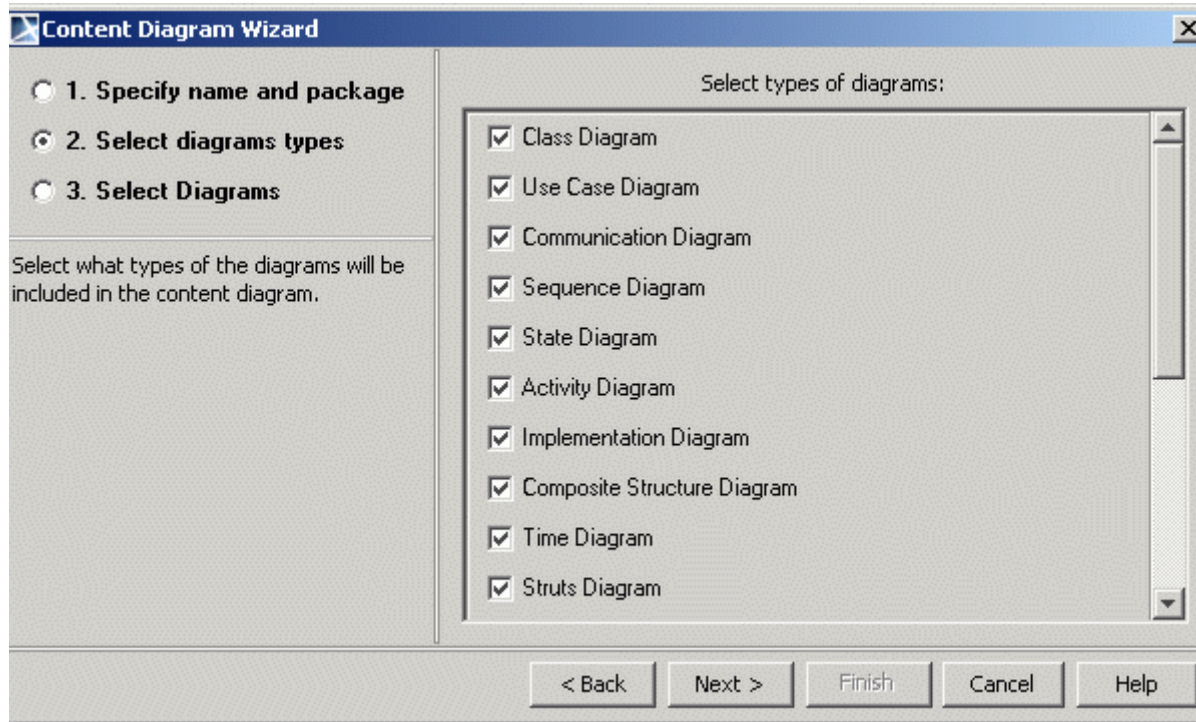


Figure 273 -- Content Diagram Wizard. Select Diagrams Types.

Select the types of diagrams to be included in the content diagram.

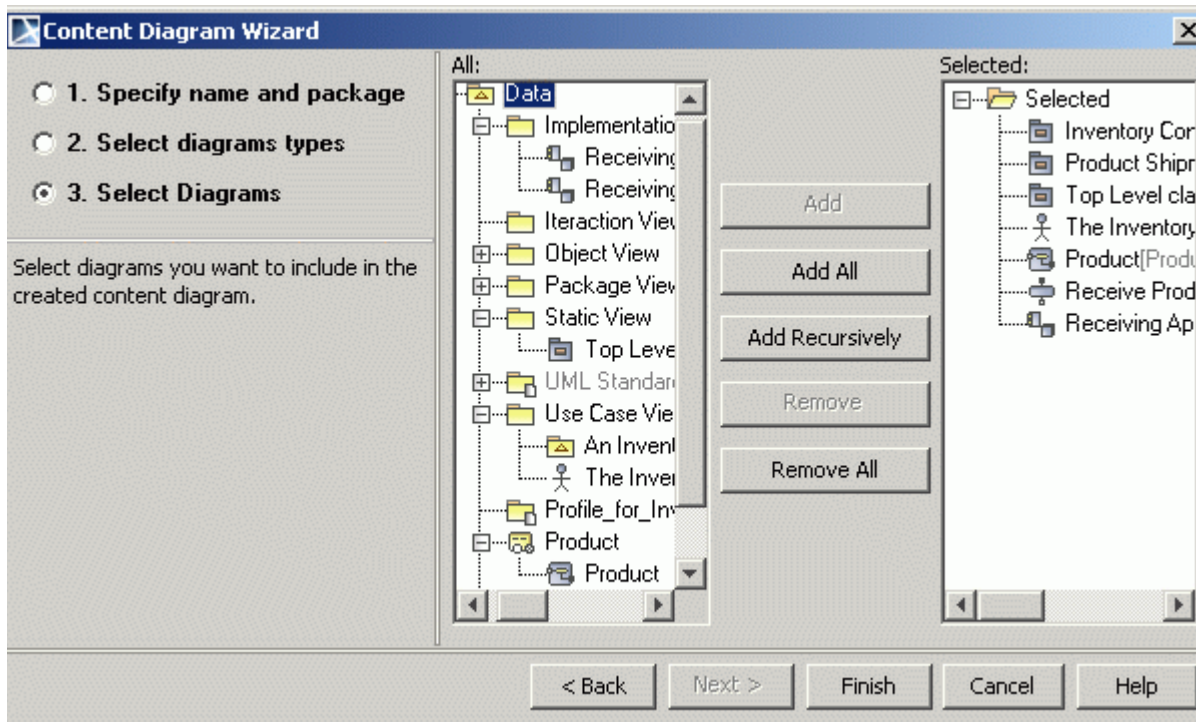


Figure 274 -- Content Diagram Wizard. Select diagrams.

Sequence Diagram from Java Source Wizard

NOTE This functionality is available in Enterprise edition only.

The **Sequence Diagram from Java Source Wizard** enables the visualization of Java method implementation with UML Sequence diagrams. Because UML Sequence diagrams cannot yet show Java code with 100% accuracy, MagicDraw provides a mechanism to generate a diagram that reflects essential Java method content.

To start the **Sequence Diagram from Java Source Wizard**

1. From the **Analyze** menu, select **Model Visualizer**. The **Model Visualizer** dialog box opens.
2. From the wizards list, select the **Sequence Diagram from Java Source Wizard**.

3. Click **Start**. The **Sequence Diagram from Java Source Wizard** opens.
 - From the **Diagrams** menu, select **Diagram Wizards**, then **Sequence Diagram from Java Source Wizard**.
 - Create a class with operation. Create a new code engineering Java set. Add a class to this set. In the Browser, Code engineering sets tree, select the RTMethod node. From its shortcut menu, select **Reverse Implementation**. The **Sequence Diagram from Java Source Wizard** opens (method is selected in the wizard).
 - Create a class with operation. Select the operation in the Browser. From the operation shortcut menu, select the **Reverse Implementation** command. The **Sequence Diagram from Java Source Wizard** opens (method is selected in the wizard).

- In the Sequence diagram, create *call* message. Select it on the diagram pane. From the message shortcut menu, select **Reverse Implementation**. The **Sequence Diagram from Java Source Wizard** opens (method is selected in the wizard).

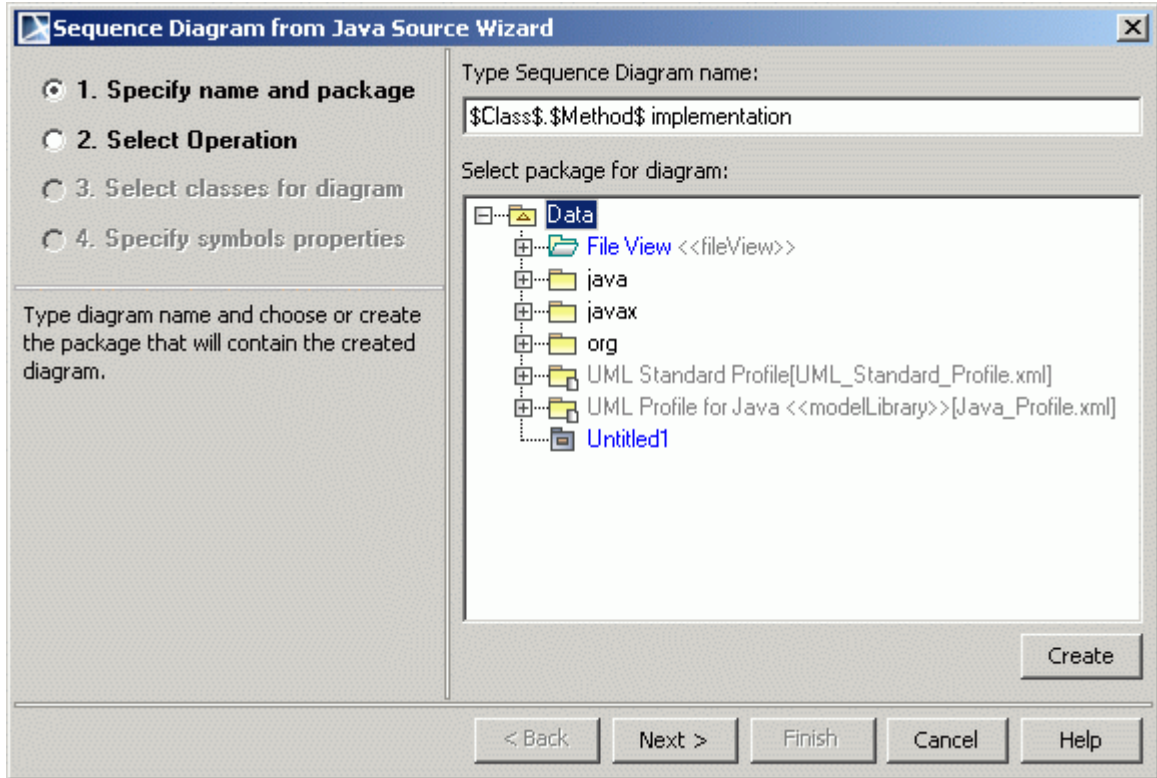


Figure 275 -- Sequence Diagram from Java Source Wizard. Specify Name and Package

In the **Type diagram name** text box, type a name for the new sequence diagram. By default, the name of the class and method is assigned to the sequence diagram.

Select the package that will contain the created diagram. The hierarchy of UML model packages is displayed in the **Select package for diagram** list window. From the **Data** tree, select the package that will be the parent for the newly created diagram, or create a new package by clicking **New**.

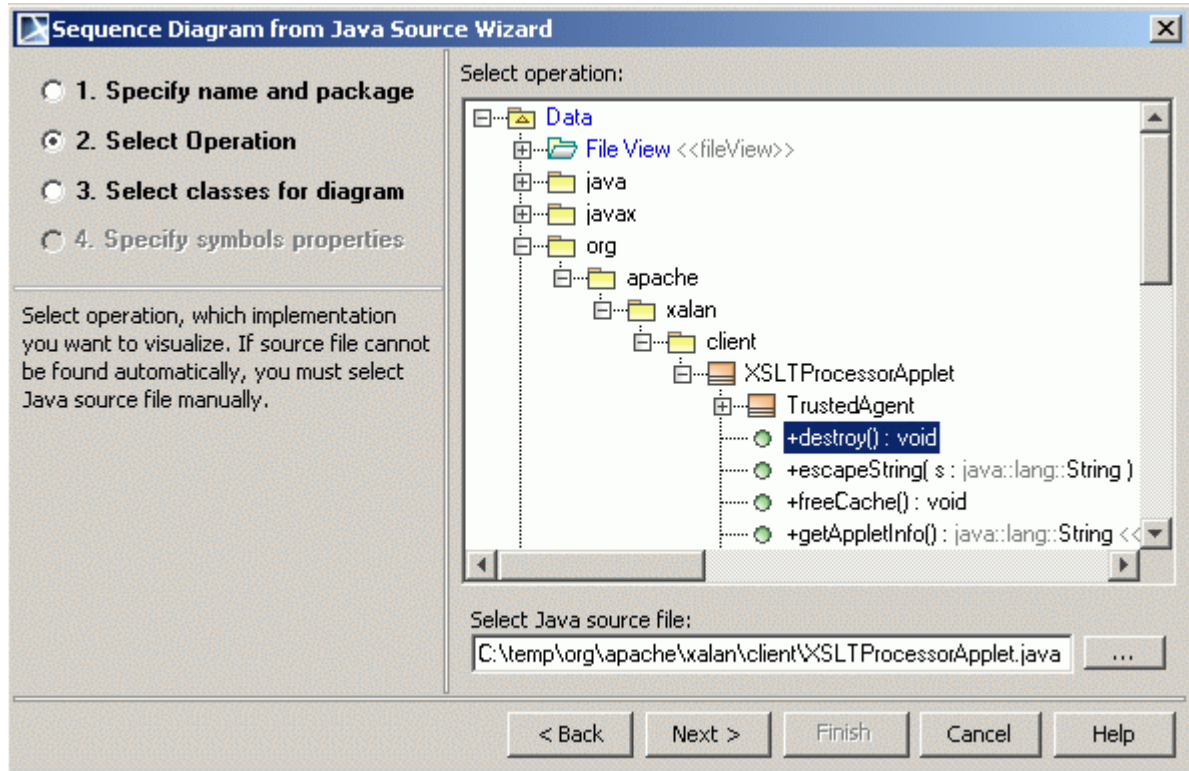


Figure 276 -- Sequence Diagram from Java Source Wizard. Select Operations.

Select the operation, which will be displayed in the sequence diagram. When you start the wizard from either the operation shortcut menu or the Sequence diagram, both the operation and Java source file are selected by default.

In the **Select Java source file field**, specify the Java source file (if this file cannot be found automatically).

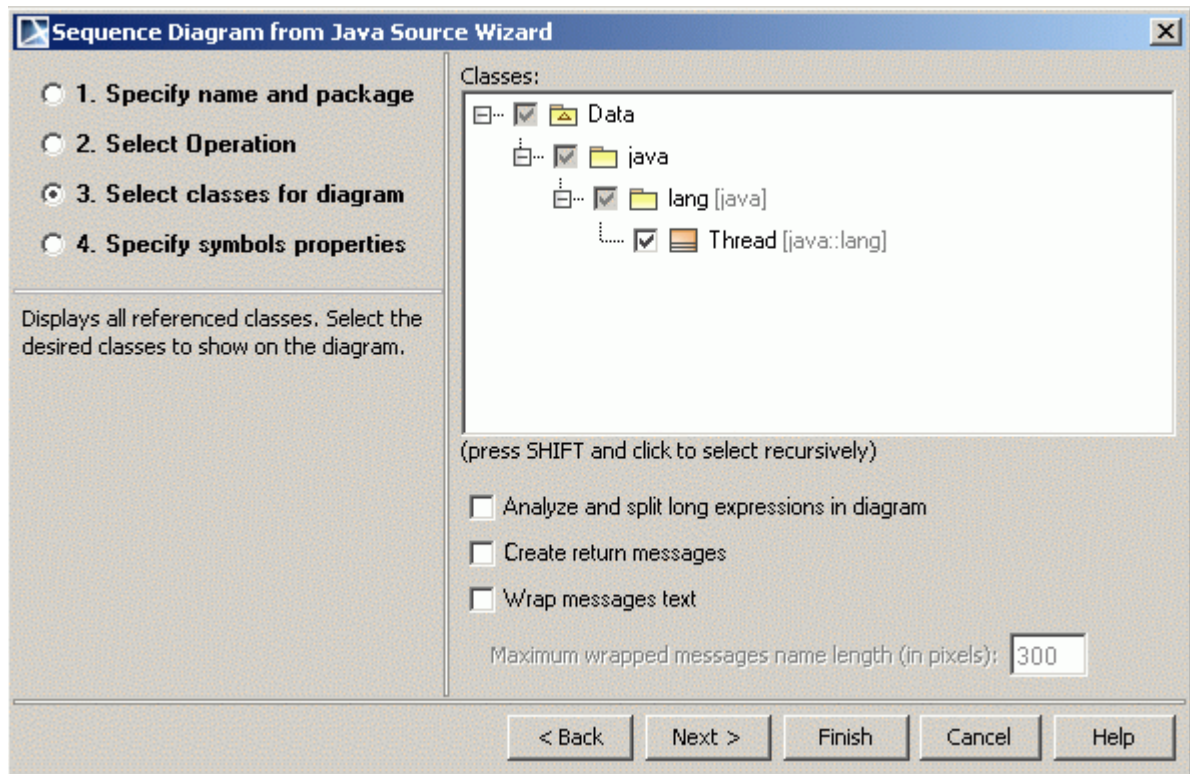


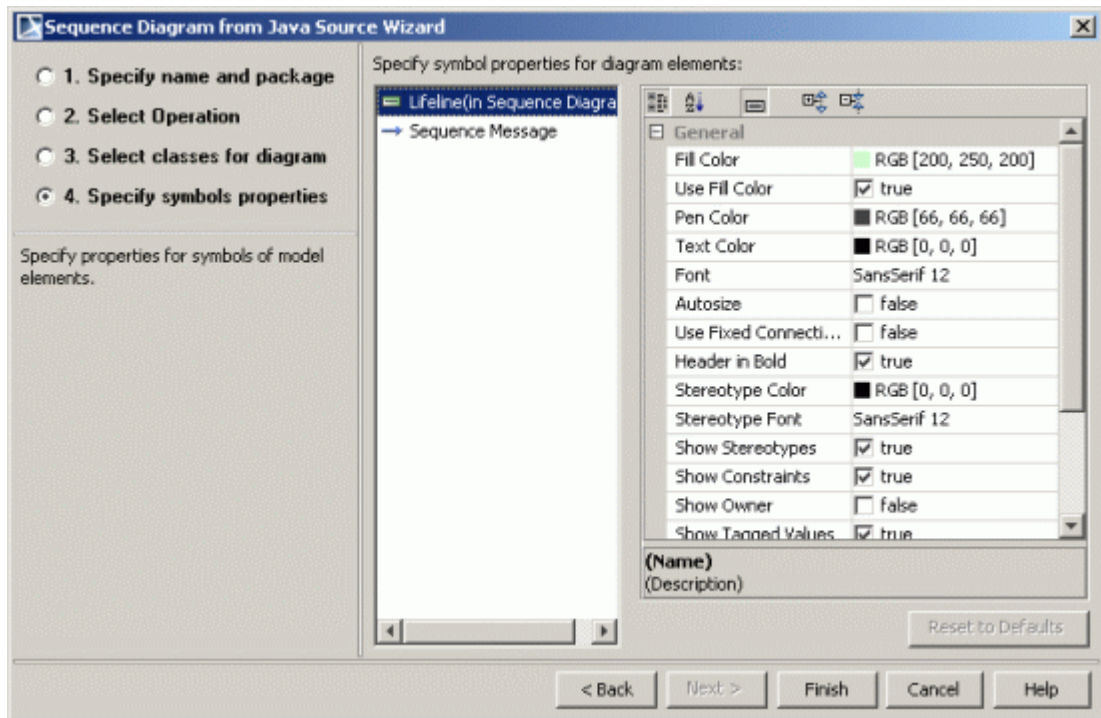
Figure 277 -- Sequence Diagram from Java Source Wizard. Select Classes for Diagram.

Select Classes for Diagram includes the following check boxes:

Analyze and split long expressions in diagram - if the expression contains calls and cannot be displayed as a call message, every call is displayed as a separate call message with temporary variable initialization. In the final expression message, these calls are replaced with appropriate temporary variable names.

Create return message - displays a return message for every call message.

Wrap message text - specifies the maximum message text length in pixels for wrapping longer messages.



Using **Specify Symbol Properties**, you can select options for the elements to be represented in your sequence diagram.

NOTE: You can extend the created sequence diagram by method: Select the method (message), which you want to be displayed with additional details (this can include referenced classes and other defining methods). Then, from the message shortcut menu, select **Reverse Implementation**. The **Sequence Diagram from Java Source Wizard** opens.

Displaying related elements

To display elements related to the selected element

1. From the **Edit** menu, **Symbol** submenu or from the shape shortcut menu, select **Related Elements** and then select **Display Related Elements**.
2. The **Display Related Elements** dialog box opens.

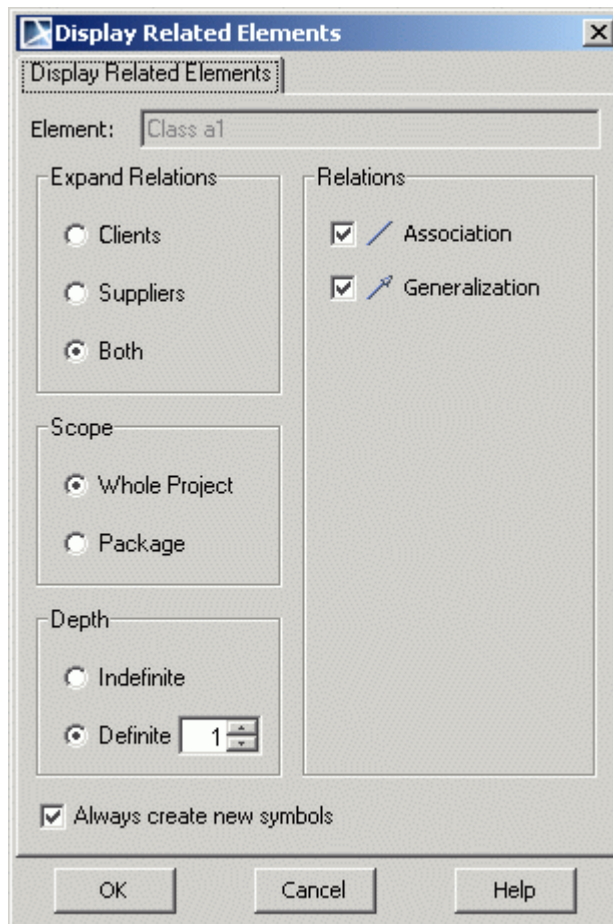


Figure 11 -- Display Related Elements dialog box

3. Select the desired options for displaying the related elements:

Option name	Function
Expand Relations	Select what kind of elements should be displayed on the diagram: <ul style="list-style-type: none">• Clients• Suppliers• Both
Scope	Select in what scope the elements related to the selected model element will be found: <ul style="list-style-type: none">• Whole Project• Package
Depth	Specify the range for searching for selected relationships: <ul style="list-style-type: none">• Indefinite. All possible relationships are involved.• Definite. Specify the level of hierarchy for the relationships involved.
Relations	Select the relationships you want to display on the diagram.
Always create new symbols	When this check box is selected, a new symbol is created even if a related element already has a symbol in the diagram.

To display paths among shapes that already exist in the model data

- From the shape shortcut menu, select **Related Elements** and then **Display Paths**.
- Select a symbol and from the **Edit** menu, select **Symbol**, then **Related Elements**, and then select **Display Paths**.

Dependencies analysis

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The MagicDraw Usages and Dependencies feature enables you to track and view element dependencies in UML models, explore how model elements are used by other elements, and understand the relationships between used and dependent elements.

It is also useful for analyzing associations between elements or searching for diagrams where these elements are represented.

To search for usages / dependent elements:

1. Select model element in the Browser or in the Diagram pane.
2. From the element shortcut menu, select the **Related Elements** command and then select either the **Used By** or **Depends On** command. The **Usages/Dependencies Search Options** dialog box opens.
3. In the **Usages/Dependencies Search Options** dialog box, specify options and click **OK**. Depending on which command you selected on the shortcut menu, either the **Element used by** or **depends on** window opens.

Usages Functionality

The **Elements used by** window provides a list of all the elements that reference the current element.

For example: If element1 references element2, then element1 uses element2. Conversely, element2 is also used by element1. In the **Attribute Specification** dialog box, add class2 in the **Type** box. This means that class2 is used by that attribute.

Containing other elements is not considered usage. For example, when a package contains an inner element class, this does not mean that the package uses the class. The class is categorized as only a container of the package.

Dependent Elements functionality

Dependency functionality gives the opposite results of those provided by the **Elements used by** window. The **Element depends on** window displays a list of the elements that depend on the current element.

For example: If element1 contains a reference to element2, then element1 depends on element2.

Usages/Dependencies Search Options dialog box

Use the **Usages/Dependencies Search Options** to specify various options (for example, search and visualization options).

The **Usages/Dependencies Search Options** dialog box covers the spectrum of usages and dependent element functionality. This means that if you clear or select any check boxes in the **Usages/Dependencies Search Options** dialog box, the next time you search for dependencies, the values for these check boxes remain the same.

To open the Usages/Dependencies Search Options dialog box:

1. Select a model element in the **Browser** tree or on the diagram pane.
2. Select **Related Elements** from the shortcut menu and then select either **Used By** or **Depends On** command.

3. The **Usages/Dependencies Search Options** dialog box opens.

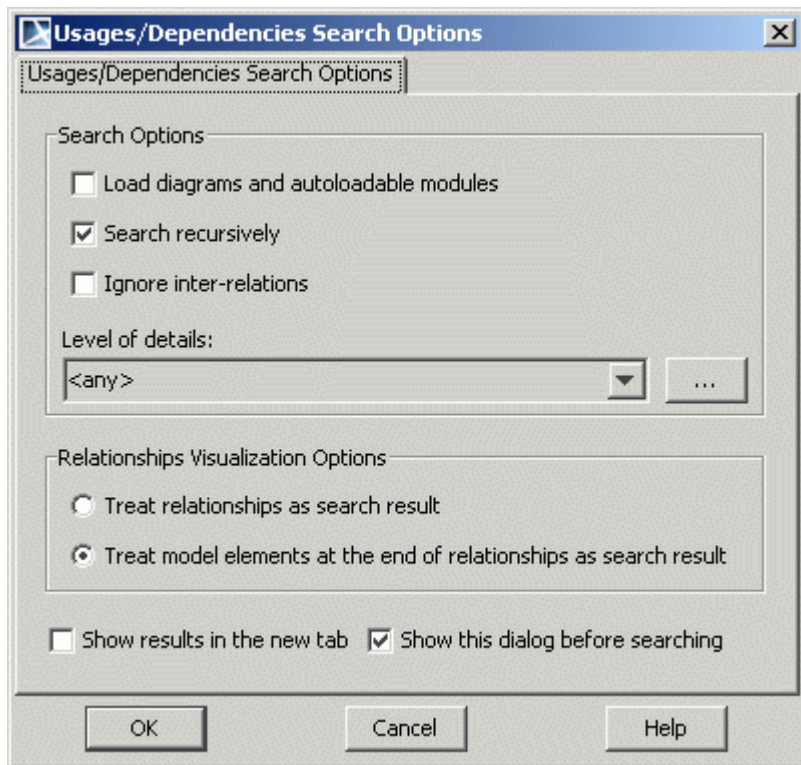


Figure 278 -- Find Usages/Dependent Elements Options

Element Name	Function
Load diagrams and autoloadable modules	If the model has unloaded diagrams or modules, select this check box to load all elements to be included in performing the usages/dependencies search.

Element Name	Function
Search recursively	<p>Usages/dependencies of inner elements (beneath the level of the current element) are listed in the search results list.</p> <p>When the Search recursively check box is cleared (default value), the usages/dependencies table lists those elements that are using the current element.</p> <p>For example: element1 contains element2. When you search non-recursively, those elements that use element1 are listed. When you search recursively, elements using element1 are listed, while the other branch lists those elements that use element2.</p>
Ignore inter-relations	<p>Only usages/dependencies outside the current element (above the level of current element and in the same level) are listed.</p> <p>You may use this option when you export the package as modules. You may analyze the dependencies of a package inner element to its outer elements (note that you cannot export a package that contains inner elements with dependencies to the outer elements according to this package). For example: create package p1 and package p2. Connect these packages with a dependency relationship. In the Browser, drag and drop this dependency to p1. In the Usages/Dependencies Search Options dialog box, select the Ignore inter-relations check box. Search for the p1 dependency. The dependency to the dependency relationship will not be found because this element is a child of p1.</p>
Level of details	<p>Select predefined types (Classifiers, Packages) from the combo box or click the "... " button, which opens Select Element/Symbol Type dialog box and allows the selection of custom types. Only elements of these types will be displayed in the search results. For example, <i>Package</i> selected as "level of details". Class B depends on Class A (owned in package p2), because it uses A as type of one operation parameter.</p> <p>In this case package p2 will be displayed as a result, class A will be added under p2 node.</p>
Treat relationships as search result	<p>Search results provide a complete list of used or dependent relationships together with other results.</p>

Element Name	Function
Treat model elements at the end of relationships as search results	Relationships are skipped and only usages/dependent elements between the model elements connected with these relationships are displayed.
Show results in the new tab	If you want new Elements Using/Dependencies windows displayed for every new search, select Show results in the new tab check box.
Show this dialog before searching	<p>If the Show this dialog before searching check box is selected, the Find Usages/Dependent Elements Options dialog box opens when you search for usages or dependencies.</p> <p>If the Show this dialog before searching check box is cleared, you may access the Find Usages/Dependent Elements Options dialog box from the Elements Using/Dependencies window by pressing the Show Usages/Dependent Elements Options button.</p>

Elements Using/Dependencies windows

The Elements Using/Dependencies window lists the results of the usages/dependencies. In the Elements Using/Dependencies window you can analyze results, search for an element location (in a diagram, for example, or in a browser), and filter results.

Because the Usages/Dependent Elements Results window is not synchronized with the model, any changes made to the model elements will not show in the results window until you click Refresh.

To open the Elements Using / Dependencies windows:

1. Select a model element in the **Browser** tree or on the diagram pane.
2. Select **Related Elements** from the shortcut menu and then select either **Find Usages** or **Find Dependent Elements**.
3. The [Find Usages/Dependent Elements dialog box](#) opens. Click **OK** to open the **Elements Using/Dependencies** window.

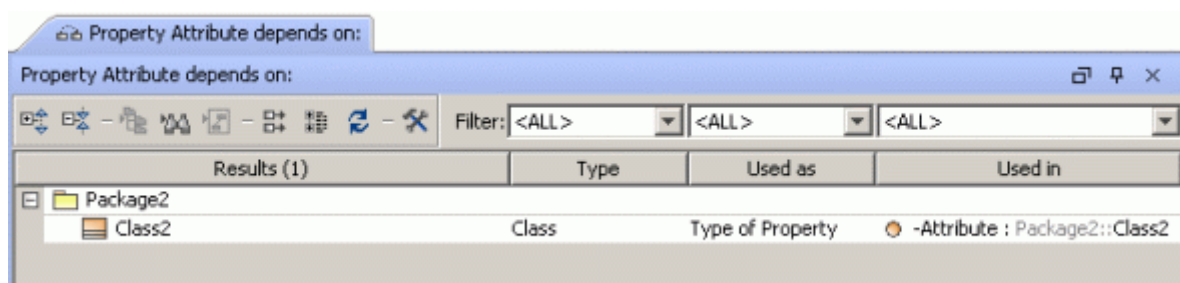


Figure 279 -- Elements using/ Dependencies window

The following table presents more information about the Elements Using/Dependencies window.

Item Name	Function
Expand	Expands records listed in groups. Click the plus sign next to the group name to display the contents.
Collapse	Collapses records listed in groups.
Select in Containment Tree	<p>In the usages/dependencies table, select the record. Click the Select in Containment Tree button. The Element is selected in the Browser, and the Containment tree, and the symbol of this element is selected in the diagram pane.</p> <p>Click the Select in Containment Tree button to open any closed and previously loaded diagrams. You can also select the element in the Containment Tree by double clicking it.</p> <p>NOTE: The Select in Containment Tree button is not available for multiple element selections.</p>
Move to Search Results	<p>In the usages/dependencies table, select the record. Click the Move to Search Results button. In the usages/dependencies table results are moved to the Browser, Search Results tree. In the Search Results tree, you will see all the results in one list and organized in two groups: From diagram and From model.</p>

Open all diagrams that contain current Usages/Dependencies	<p>Diagrams, which are referenced in the usages/dependencies table, are opened. In the open diagrams, the view is focused on used/dependent elements.</p> <p>NOTE: The Open all diagrams that contain current Usages/Dependencies button is inactive when there are no elements that are used in diagrams.</p>
Show/hide the Full Path Names	<p>The full path is displayed next to the element name.</p> <p>NOTE: For a symbol this button is not valid.</p>
Refresh	<p>The usages/dependencies table should be refreshed after:</p> <ul style="list-style-type: none">• Elements or symbols are deleted.• New dependencies/usages are created for the particular model element.
Show Usages/Dependencies Search Options	<p>The Find Usages/Dependent Elements Options dialog box opens.</p>
Filter	<p>Filter boxes in the Elements used by and depends on windows are placed above the column, which will be filtered.</p> <p>In the used by window, you can filter by element type or usage type. In the depends on window, you can filter according to element type and dependency type. When you expand the Filter box, you will see items listed in the usages/dependencies columns.</p> <p>NOTE: In the Element Type Filter box within the used by window, combo box, additional Show All but Symbols filtering options is added.</p>

Model Differencing

View Online Demo Visual Model Differencing

NOTE This functionality is available in Architect, and Enterprise editions only.

The use of model differencing functionality allows you to compare two MagicDraw UML local projects or teamwork project versions, as well as diagrams. Model elements are compared by the element ID.

You can compare:

- current project with locally saved project.
- current project with open project.
- two teamwork project versions.
- current project with teamwork project version.
- local project and teamwork project version.
- diagrams.

Models comparison

First of all you have to select the projects you want to compare. Model comparison will be displayed as two model trees. Differences between the model versions are marked using colors and highlighting.

When comparing two models the following data changes will be reflected:

- New model elements.
- Deleted model elements.
- Model elements with modified data.
- Model elements that changed location.
- Inner elements changes.

To select the projects you want to compare

1. Open a project you want to compare and from the **Analyze** menu, select **Compare Projects**.
2. The **Compare Projects** dialog box opens.

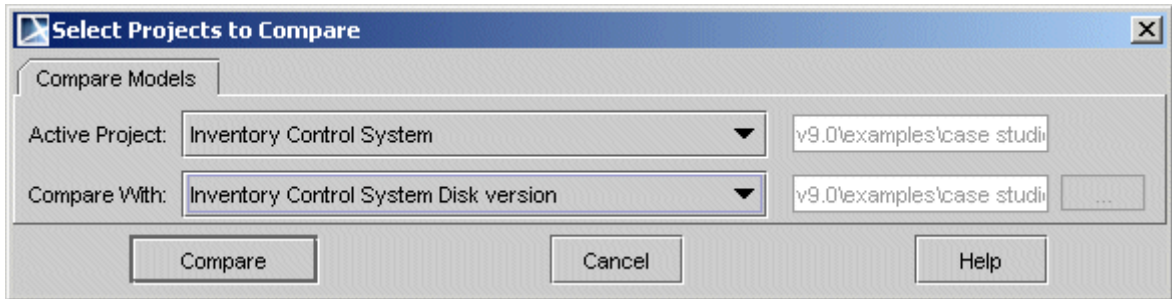


Figure 280 -- Select Projects to Compare dialog box

3. From the **Active Project** combo box, select a project you want to compare with another project. The list contains names of all open projects in MagicDraw.
4. From the **Compare With** combo box, select a project with which you want to compare the first project. The following options are available:
 - open project names are listed, except the project, which is selected in the **Active Project** combo box.
 - **Local Project**. Click the "..." button and select a project version you want to compare the current project with.
 - **Teamwork Project**. Click the "..." button and from the Open Teamwork Project dialog box, select a project and its version you want to compare with.
 - **<current project>Disk version**. By selecting this option you will be able to compare currently changed project version with the unchanged project version on the disc without current changes.

By default the first opened project should be selected. If there are no more projects open, Local Project is selected as the value.

Understanding model differences

All model version differences are displayed in the **Difference Viewer** dialog box.

8 MODEL ANALYSIS

Model Differencing

Difference Viewer

Display: All Differences

Control System_Disk version C:\Pr...y Control System_.xml.zip*

Data

- Implementation View
- Interaction View
- Object View
 - 38:Static View::VendorProduct
 - 426:Static View::CustomProduct
 - 4321:Static View::Shipment
 - 47:Static View::VendorProduct
- Product Shipment object diagram
- Package View
- Purchasing
- Inventory Control System package diagram
- Index

Data

- General View
 - Implementation View
 - Package View
 - Purchasing
 - Customer
 - Salesman
- Interaction View
- Inventory Control System package diagram
- Index

Export Differences Compare Diag

Property	Inven...trol System_Disk version	C:\Pr...y Control System_.xml.zip*
Package	Package View	Package View
Abstract	<input type="checkbox"/> False	<input type="checkbox"/> False
Abstract	<input type="checkbox"/> False	<input type="checkbox"/> False
Abstract	<input type="checkbox"/> False	<input type="checkbox"/> False
Documentation		
Stereotypes		
Named Values		
Constraints		
Active hyperlink		
Owner	Data	General View

 Inserted
 Deleted
 Modified
 Modified inner

Differences found: 3 inserted elements, 9 deleted elements, 5 modified elements

Close Help








Figure 281 -- Difference Viewer dialog box

In the **Difference Viewer** dialog box two compared projects are displayed. The number of differences is displayed at the bottom of the dialog box. The number of differences number includes inserted, deleted, and modified elements. The number does not include elements with changes to inner elements. Differences number is displayed in following order:

To mark changes in the model elements several colors are used:

- *Elements that do not exist in the other model (inserted elements).* Element is displayed only in the right-hand tree. New element is highlighted in light green.
- *Elements that exist in the other model, but do not exist in current model (deleted elements).* Elements are always displayed in the left-hand tree. The deleted element is highlighted in grey.
- *Elements with modified element data (modified elements).* The modified element is highlighted as changed in both trees. The modified element is highlighted in light blue.
- *Elements that changed location (parent has changed).* Element is marked as a modified element. Empty nodes are displayed in the opposite tree where the element does not exist. On the moved element and on its former position, a button is displayed. Pressing the button on the former position, selects the place where the element has moved to. Pressing the button on the moved element position, selects the former element place. Also, you can perform these operations using the shortcut menu commands **Go to former position** or **Go to moved element**.
- *Elements that have inner elements that changed.* The element is marked in both trees and is highlighted in light grey dashes. An element with modified element data and changed inner elements is marked as modified and as element with changed inner elements. Element is marked as modified when:
 - Element specification properties have changed. Element specification properties include all properties, which are not displayed in the Browser as separate elements. Model element specification properties are treated as changed only if the element property can be changed from the element specification. If an element specification has changed because of changes made to other elements, the element should not be treated as changed. Example: typed values in the tagged values specification, attribute links in an object and instance specification, etc.
 - Element parent has changed.
 - Path is drawn from/to element.

Buttons available in the **Difference Viewer** dialog box:

Button	Function
 Expand All	Both trees are expanded.
 Collapse All	Both trees are collapsed.
 Go to Previous Difference	Select the difference listed prior to the current one.
 Go to Next Difference	Select the difference listed next from the current one.
 Filter	The Filter dialog box opens. Show/hide the elements you want to analyze.
 Include Relation Ends	Relations are displayed in the elements specifications. A relation added to the element means that the element is marked as modified.
Display:	All - Shows all elements of the projects.
	All Differences - Shows only differences made between the projects.
	Deleted Elements - Shows only elements that were deleted from the projects.
	Inserted Elements - Shows only elements that were inserted in the projects.
	Modified Elements - Shows only elements that were modified.
 Find Next Element	The Find dialog box opens. Search for elements within the corresponding project.
More>> <<Less	More/Less button shows/hides the element properties table. By default, the properties table is hidden.

Button	Function
Export Differences	<p>Creates .html or .txt differences report. In the Save Difference Report As dialog box, select the directory where you want to save this report.</p> <p>NOTE: *.html format is suitable for viewing the difference report. If you want to copy this report to another program, use of the *.txt format is recommended.</p>

The first column of the Property window contains the same properties as the Quick Properties tab in the Browser. The second column title is the left-hand project name (with path) for local projects, and teamwork project name and version number for teamwork projects. The third column title is the right-hand project name (with path) for local projects, and teamwork project name and version number for teamwork projects. Modified properties are marked with the same color as in the model element tree.

Diagrams Comparison

To compare diagrams

1. In the **Difference Viewer** dialog box, select the diagram you want to compare and click the **Compare Diagrams** button.

2. The **Diagrams Difference Viewer** dialog box opens, which displays both diagrams with changes made in them.

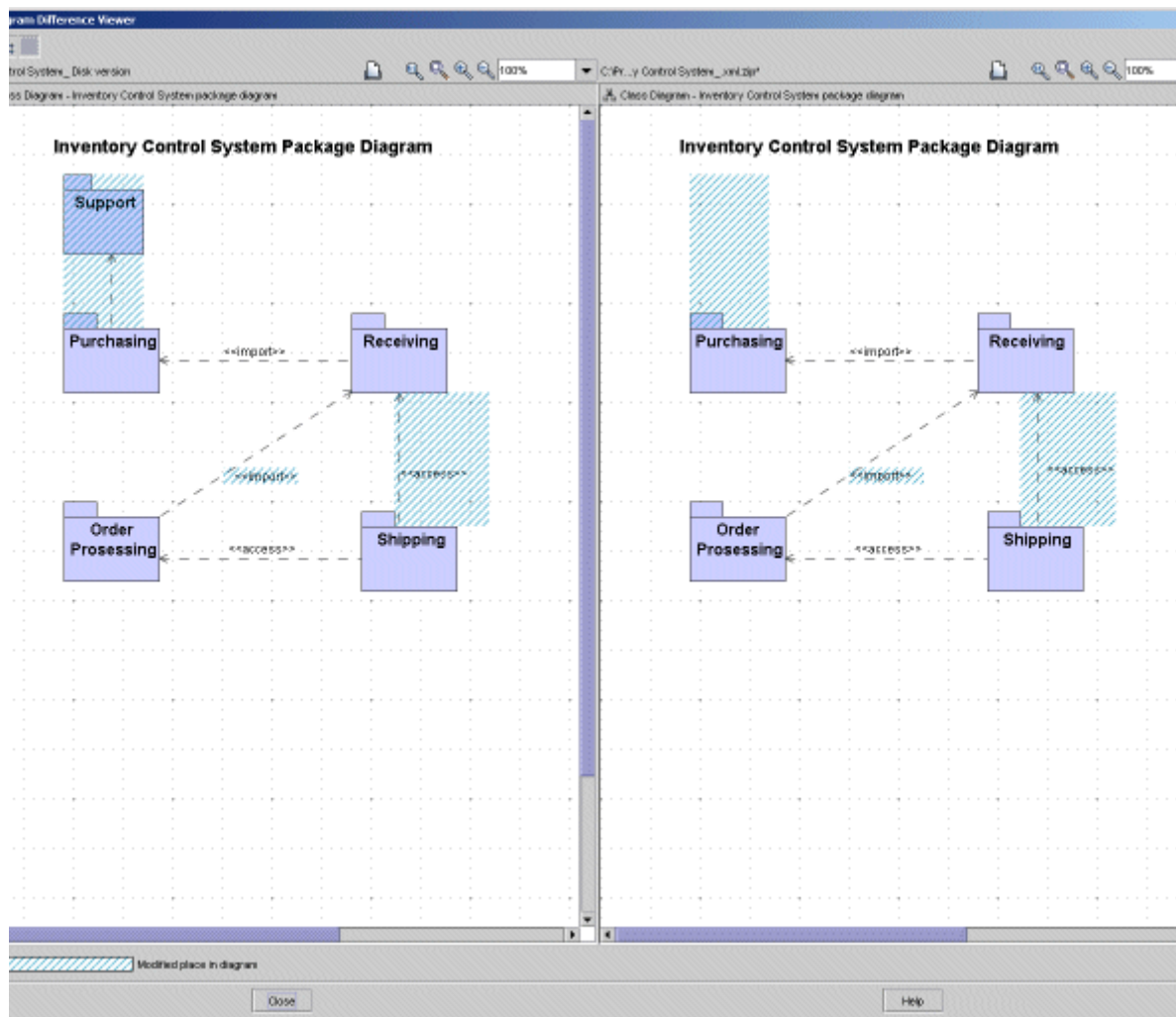


Figure 282 -- Diagram Difference Viewer dialog box

Diagrams Difference Viewer displays two diagrams:







- Current (or first opened) project diagram is displayed at the right-hand side.
- Diagram that is compared with is displayed at the left-hand side.




Symbol changes that are reflected in the diagram:

- Modified symbol properties
- New symbol creation
- Symbol deletion
- Symbol bound changes (resize, bound changes because of element properties changes)

All changes are highlighted in light blue dashes.

Buttons available in the **Diagrams Difference Viewer** dialog box:

Button	Function
 Synchronize Zooming	If pressed, zooming affects both diagrams: zooming one diagram causes zooming of the other diagram.
 Synchronize Scrolling	If pressed, scrolling one diagram causes scrolling of the other diagram.
 Mark Changes	If pressed, places where diagram has changed are marked on both diagrams.
 Print Diagram	Prints the corresponding diagram together with marked changes.
 Zoom 1:1	Zooms the corresponding diagram(s) to the original size.
 Fit in Window	Zooms the corresponding diagram(s) to the size that fits the window.

Button	Function
 Zoom In	Zooms the corresponding diagram(s) in.
 Zoom Out	Zooms the corresponding diagram(s) out.
	Select the percentage for zooming the corresponding diagram(s).

Metrics

NOTE This functionality is available in Architect and Enterprise editions only.

MagicDraw metrics functionality, which is implemented as a plugin and accessible through the Open API, enables you to measure your project from three different viewpoints:

- UML model metrics
- System metrics
- Requirements metrics.

Using UML model metrics, you can measure your project using package, class, and diagram measurements (for example, measuring the number of classes, inheritance tree depth, and so on).

System metrics analyze models using the most popular object oriented project metrics: Halstead, McCabe, Chidamber, and Kemerer defined metrics (for example, cyclomatic complexity and weighted methods per class).

Requirement metrics consist of function points and use case metrics. These two metrics groups are so structurally similar that use case metrics are regarded as a subset of function point metrics. Use case metrics measure both the number of use cases in a project and the user case analysis through selected tagged values (priority, for example).

The results of these analyses are displayed in a table, where you can select which metric you would like displayed. You can also export the metrics to a separate file.

Metrics are implemented as a plugin and are accessible through the Open API.

A metric is a numeric value that measures a model or is counted according to model measuring. Each metric has both a lowest and highest limit specified. Metrics that fall outside of this range are marked:

- Values that are too low are displayed in a blue font.
- Values that are too high are displayed in a red font. **Note:** if the highest limit equals zero, the metric is never marked as too high.

Metric Suites

In MagicDraw, you can create your own metric suites or use one of the three predefined metric suites: *System Metrics*, *UML Model Metrics*, or *Requirements Metrics*.

The metric suites contain a list of metrics that will be counted and the properties specified for each selected metric.

To create your own metric suites, clone an existing suite and specify the suite properties. You can edit the predefined metrics suites, and all metric suites can be imported or exported, facilitating the exchange of ideas with other users.

Displaying Metrics

Metrics are counted according to properties defined in a selected metric suite and can be counted for an entire project or just the selected packages, classes, interfaces, or diagrams. The results are displayed in the Metrics window, which opens at the bottom of the MagicDraw application window.

The Metrics window contains two tabs:

- **Data** tab. The counted metrics of the selected suite are displayed in a metrics table, which includes counted metrics only.
- **Graphics** tab. The selected metric is displayed as a graphic.

Metrics tables display packages, classes, interfaces, and diagrams. Additionally, elements that contain packages, classes, interfaces, and diagrams, which are displayed using a tree structure, are not counted for these elements.

The following is an example of a metrics table structure:

Model Element	Metric1	Metric2	Metric3	MetricN
PackageA	value	value	value	value	value	value
Inner class1	value	value	value	value	value	value
Inner class2	value	value	value	value	value	value

If a value is not counted for a class, interface, package, or diagram, the cell is left empty.

You can apply the following filters to the metrics table:

- All
- Packages
- Classes (classes and interfaces are displayed)
- Diagrams
- Package Violations (only rows that contain package violations are displayed)
- Class Violations (only rows that contain class or interface violations are displayed)

When the Classes, Diagrams, or Class Violations filters are selected, the owner is displayed next to the following element: c1 (Classes::Package1)

Starting Metrics

To start Metrics:

Click **Metrics** on the **Analyze** menu, or on the class/package/interface/diagram shortcut menu, click **Tools** and then click **Metrics**.

The **Metrics** dialog box opens.

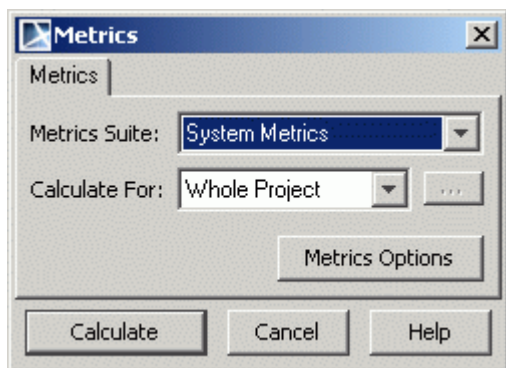


Figure 283 -- Metrics dialog box

Element	Description
Metrics Suite	Lists all the available metrics suites.
Calculate For	Lists two values: <ul style="list-style-type: none"> • Whole Project – calculates metrics for the entire project. • Selection – calculates metrics for selected items only. Click the "... " button to open the Select Elements dialog box.
Metrics Options	Opens the Metrics Options dialog box.
Calculate	Opens the Metrics window.

To set element options for Metrics, click the "..." button to open the **Select Elements** dialog box:

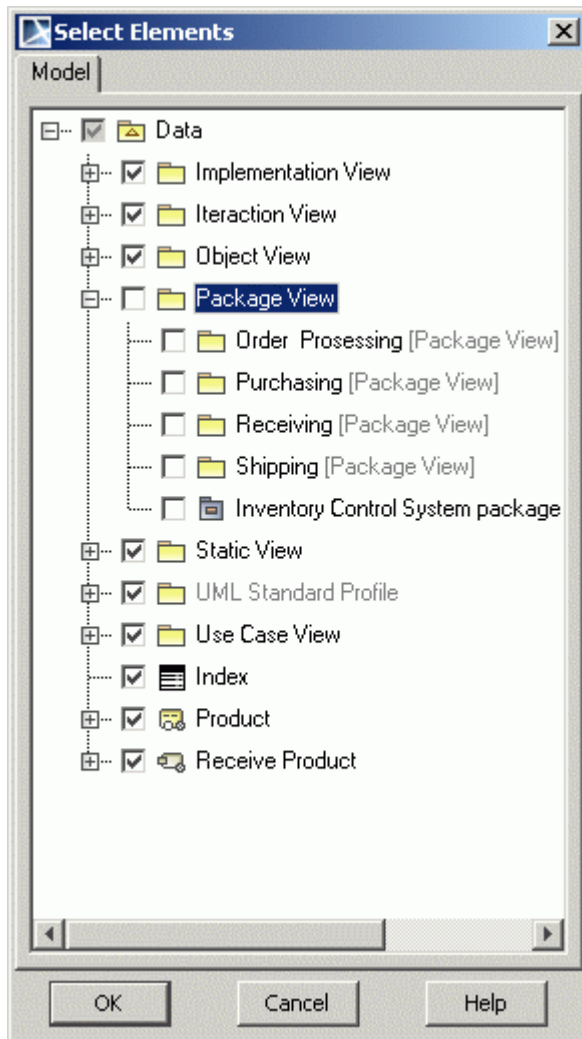


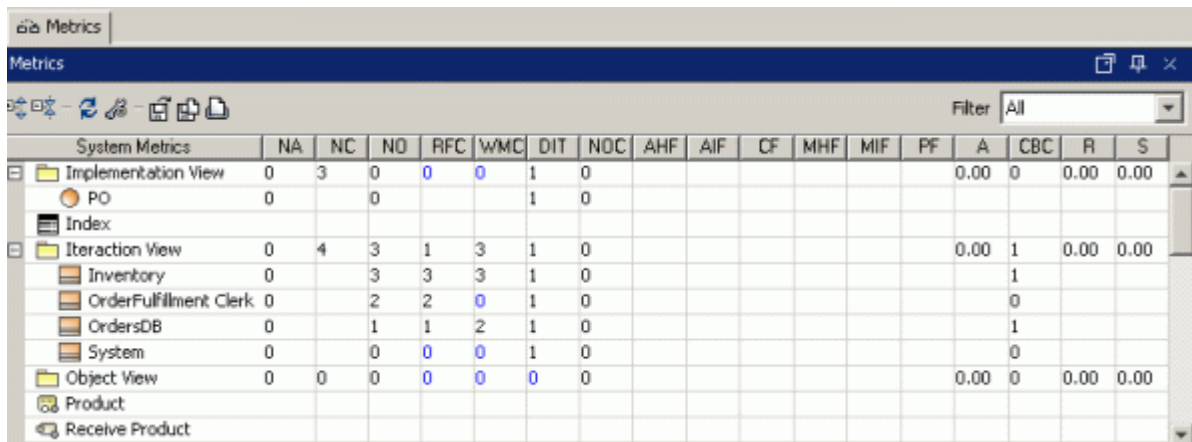
Figure 284 -- Select Elements dialog box

Packages, classes and diagrams are displayed in the **Select Elements** dialog box. If you select the box next to a parent element (for example, the *Data* check box in the image above), all its related child elements are automatically selected. Conversely, clearing the box next to a parent element clears all its related child elements.

If you clear the box next to a child element, the parent box is also cleared. For example, if the *Data* box is selected, all its related child elements are selected. If you then clear the *Package View* box, its child elements are also cleared, as is the box next to *Data*, but all the other boxes remain selected.

Metrics window

The **Metrics** window is implemented as a JIDE GUI window. Like the **Messages** window, it is available at the bottom of the MagicDraw application window.



The screenshot shows the 'Metrics' window with a table of metrics. The table has columns for various metrics: NA, NC, NO, RFC, WMC, DIT, NDC, AHF, AIF, CF, MHF, MIF, PF, A, CBC, R, S. The rows represent different system elements, including Implementation View, PO, Index, Interaction View, Inventory, OrderFulfillment Clerk, OrdersDB, System, Object View, Product, and Receive Product. Some cells contain numerical values, while others are empty or contain zero.

	NA	NC	NO	RFC	WMC	DIT	NDC	AHF	AIF	CF	MHF	MIF	PF	A	CBC	R	S
Implementation View	0	3	0	0	0	1	0							0.00	0	0.00	0.00
PO	0		0			1	0										
Index																	
Interaction View	0	4	3	1	3	1	0							0.00	1	0.00	0.00
Inventory	0		3	3	3	1	0								1		
OrderFulfillment Clerk	0		2	2	0	1	0								0		
OrdersDB	0		1	1	2	1	0								1		
System	0		0	0	0	1	0								0		
Object View	0	0	0	0	0	0	0							0.00	0	0.00	0.00
Product																	
Receive Product																	

Figure 285 -- Metrics window

GUI element	Description
Expand Current Branch	Expands all elements in the selected branch within the results table.
Collapse Current Branch	Collapses all elements in the selected branch within the results table.
Refresh	Recalculates metrics results according to the current model.
Metrics Options	Opens the Metrics Options dialog box.
Export	Opens the Export Metrics dialog box.
Compare Metrics With...	Opens the Open dialog box, where you can select a text file to compare with the currently open metric set.
Print	Prints the metrics table. The Print dialog opens.

Filter	<p>Contains these values:</p> <ul style="list-style-type: none"> • All • Packages • Classes • Diagrams • Package Violations • Class Violations
--------	--

The selected metrics rows or cells can be copied to the clipboard by clicking **Copy** on the shortcut menu or by **Ctrl+C** on your keyboard.

Exporting Metrics

You can export the selected metrics rows and columns, or the entire metrics table, to a metrics results file. Metrics results can be exported using *.txt and *.html formats.

In the following example, metrics are presented in *.txt format and are separated by tabs:

Element	Metric1	Metric2	Metric3	MetricN
Package Package1	value	value	value	Value
Class class1 (Package1::class1)	value	value	value	Value
Class class2 (Package2::class2)	value	value	value	Value

Here *Metric1 .. MetricN* – the metric name abbreviation.

Technical information is displayed at the bottom of the file. Text "Element IDs" are added after the metrics of an element and are also printed. This information is needed for metrics comparison.

In the following example, information is presented in *.html format:

Metrics Report

Element	<u>Metric1</u>	<u>Metric2</u>	<u>Metric3</u>	<u>Metric N</u>
Package Package1 (<i>Package1</i>)	value	value	value	Value
Class class1 (<i>Package1::class1</i>)	value	value	value	value

Here *Metric1* .. *MetricN* – the metric name abbreviation.

Each metric name is hyperlinked with its metric description. Metric descriptions can be opened in a separate window after clicking the hyperlink.

NOTE: *.html format is best suited for viewing metrics. If you want to copy the metrics table to another program, use of the *.txt format.

To export **Metrics**, click the **Export Metrics** button in the **Metrics** window. The **Export Metrics** dialog box opens:

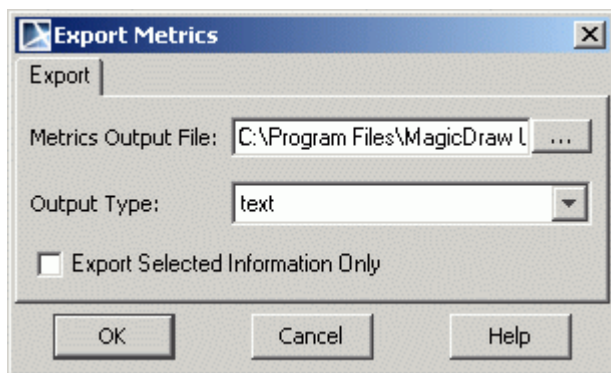


Figure 286 -- Export Metrics dialog box

Element	Description
Metrics Output File:	Displays the path and file names of the metrics results output file. Click the "... " button to select the location and file.

Output Type:	Contains these values: <ul style="list-style-type: none"> • Text (*.txt) • HTML (*.html)
Export Selected Rows Only	When selected, only the selected table rows and the header row are exported. When cleared, the entire metrics table is exported.

Comparing metrics

Counted metrics can be compared with metrics that are saved in a *.txt file. Metrics can be compared only when the metrics window is open.

Comparison results are displayed in the same metrics table. If a cell contains a metric that has increased, it has a red fill color. If the metric has decreased, a blue fill color is used. Metrics that are not found in other file cells have a grey fill color.

The metrics comparison can be canceled using the ESC key.

Metrics Options

Metrics suites are managed in the **Metrics Options** dialog box.

To open the **Metrics Options** dialog box:

- From the **Analyze** menu, select **Metrics** and then **Metrics Options**
- In the Metrics dialog box, click **Metrics Options**.

The left pane of the **Metrics Options** dialog box displays the defined metrics suites. Using the buttons or shortcut menu, metrics suites can be cloned, renamed, removed, exported, and imported. Predefined metrics sets cannot be renamed or removed.

The suite properties are displayed in the right pane:

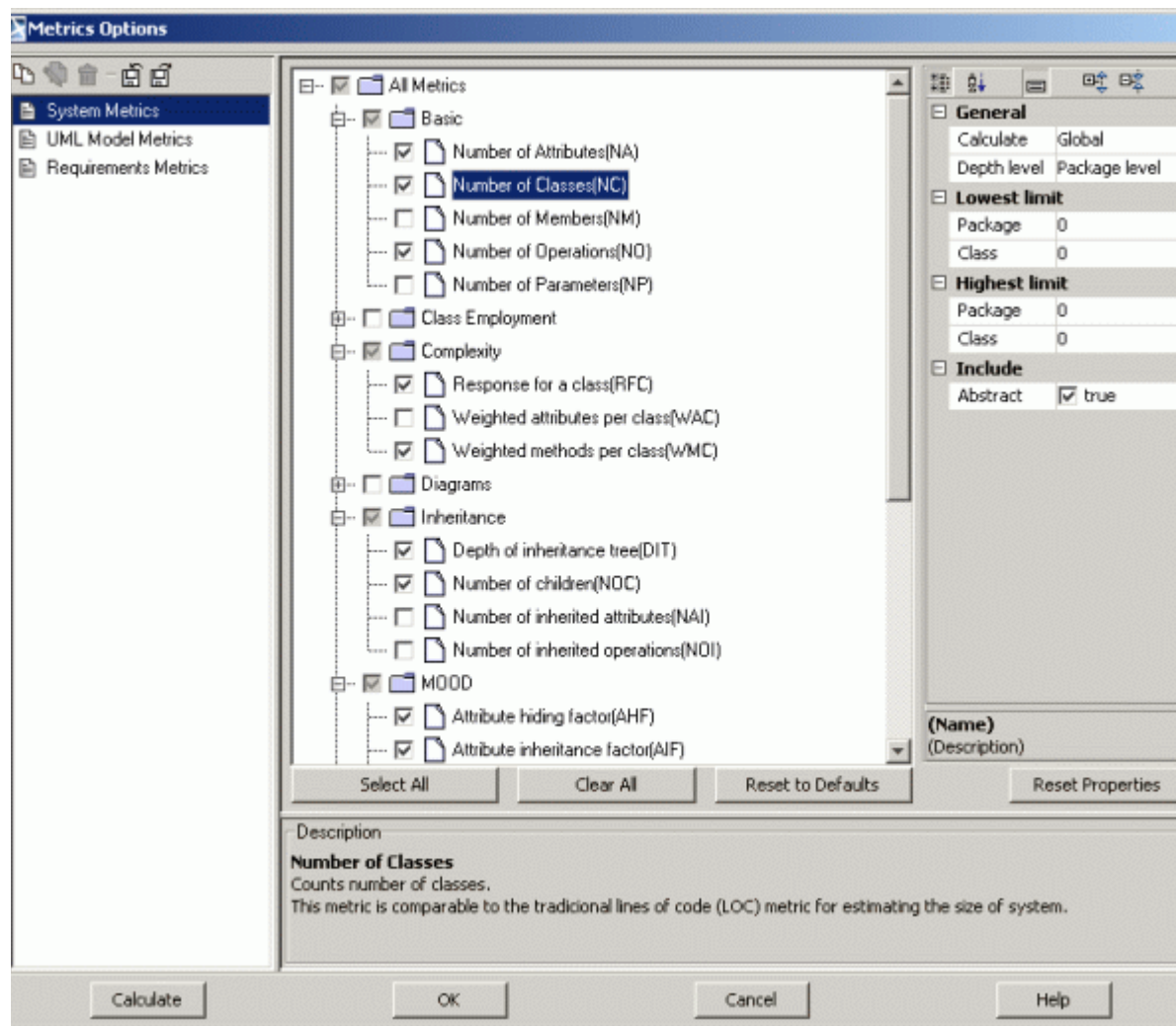
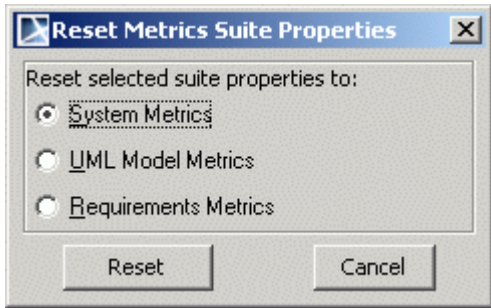


Figure 287 -- Metrics Options dialog box

Element	Description
Metrics suites list	Displays all created metrics suites in a list.

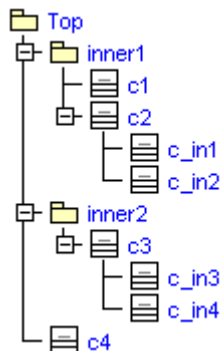
Metrics suites list buttons: Clone Rename Remove Import Export	<ul style="list-style-type: none"> • Clone – clone the selected suite. • Rename – rename the selected suite. • Remove – remove the selected suite. • Import – import a new suite. The Open dialog box opens. • Export – export the selected suite. The Save dialog box opens. <p>All these commands are available from each metric suite shortcut menu.</p>
Metrics tree	Use this tree to select the metrics you want to include in your metrics suite. All metrics are displayed in the metrics tree.
Properties list	Metrics properties are displayed individually for each property.
Select All	Selects all metrics.
Clear All	Clears all metrics.
Reset to Default	<p>Predefined MagicDraw metrics are reset to the default metrics suite.</p> <p>User-created metrics suites are reset to the selected predefined metrics suite. The Reset Metrics Suite Properties dialog box opens:</p> 
Description	Displays the selected metric description.
OK	Saves all changes and closes the dialog box.

Metrics Properties

Properties Group	Property	Description
General	Calculate	Defines what will be counted: <ul style="list-style-type: none"> • Local – inside package (class). • Global - inside package (class) recursively. • Average – metrics will be counted from the lowest level of the elements tree. Each upper level metric will be counted as an average of the current object metric and all lower level metrics: $\text{Average_element_metric} = (\text{Element_metric_value (if counted separately)} + \text{sum (inner_elements_metrics_values)}) / (1 \text{ (if element_metric_value was counted)} + \text{count_of_inner_elements_that_have_metrics_counted})$ Average metric value should be rounded down to the lower value (for example, 1.5 = 1, 1.6 =2) • Min – lowest level metrics will be counted. Each upper level metric will be set to the minimum of the current object metric and all lower level metrics (except the metrics that are equal 0). $\text{Min_element_metric} = \min (\text{Element_metric_value}, \min (\text{inner_elements_metrics_values}))$ Here metric_value > 0 • Max – lowest level metrics will be counted. Each upper level metric will be set to the maximum of the current object metric and all lower level metrics. $\text{Max_element_metric} = \max (\text{Element_metric_value}, \max (\text{inner_elements_metrics_values}))$ Here metric_value > 0
Lowest limit	Package	Recommended lowest metric value for the package. Editable.
	Class	Recommended lowest metric value for class and interface. Editable.
	Diagram	Recommended lowest metric value for the diagram. Editable.

Properties Group	Property	Description
Highest limit	Package	Recommended highest metric value for package. Editable. Note: if the highest limit is equal to 0, the metric is never marked as too high (in red font color).
	Class	Recommended highest metric value for class and interface. Editable. Note: if the highest limit is equal to 0, the metric is never marked as too high (in red font color).
	Diagram	Recommended highest metric value for diagram. Editable. Note: if the highest limit is equal to 0, the metric is never marked as too high (in red font color).
Include		This properties group specifies whether the information is included when counting metrics.
Weight		This properties group specifies whether the information is included when counting metrics.











The following is an example of a metrics calculation used for calculating the number of classes (NC) in this tree:



Calculated metric values with a different aggregation:

Element	Local	Global	Average	Min	Max
Top	1	8	1	1	2

Calculated metric values with a different aggregation:

 Inner1	2	4	1	2	2
 C1	0	0	0	0	0
 C2	2	2	1	2	2
 c_in1	0	0	0	0	0
 c_in2	0	0	0	0	0
 Inner2	1	3	1	1	2
 C3	2	2	1	2	2
 c_in3	0	0	0	0	0
 c_in4	0	0	0	0	0
 C4	0	0	0	0	0

Dependency Matrix

[View Online Demo](#) Dependency Matrix

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The Dependency Matrix is a method of visualizing and representing dependency criteria. Diagrams, UML, and extended UML elements serve as row and columns entries. The cells in the matrix show where these elements are associated - related.

Dependency matrixes include different dependency criteria: UML relations, extended UML relations, semantic dependencies (dependency through property), and relationships through tags.

Relationship through tags is a relationship where a cell represents a relation that is implemented as a tag added to the element with a reference to another element. A relation through tags allows relate

UML element of any type. Tags are one of the methods for relating elements that cannot be represented on the same diagram.

The Dependency Matrix fulfills the feature, which helps visualize the many-to-many traceability from elements not from the same diagrams. The Dependency Matrix also provides the visualization of many-to-many for large interconnected system elements. The most usefulness is complete the lack of functionality to support different domains - DoDAF.

Dependency Matrix functionality is useful for:

- Quickly visualizing dependency criteria.
- Compactly visualizing relations of a big system. Such system relations cannot be represented by a diagram on a single sheet of paper, as the diagram is very big.
- Creating a matrix template for domains and supporting domain specific element relation visualizations.
- Studying relations from a particular scope and type of element by filtering the unimportant.
- Showing relations that cannot be represented in diagrams: representation (class by lifeline); behavior representation in other diagrams, operation representation by Call Behavior Action, Use Case relations with describing activities through property Owned Behavior, etc. The Semantic dependency matrix is needed for deeper model analysis. The Matrix allows the representation of any kind of relations through the element property.
- Another method of showing custom relations - through tags.

Creating the Dependency Matrix


The matrix element in the model is similar to the diagram element. After creating a new matrix, it appears in the Browser as a model element and the matrix pane can be opened by double-clicking on the matrix name. The same actions, which can be performed with diagrams, are valid for matrixes.

To open the Matrix Dependency View

- From the **Diagrams** main menu, select **Dependency Matrixes**. The **Dependency Matrixes** dialog box opens. Click **Add** to create a new dependency matrix and then click **Open**.

- From the **Analyze** main menu, select **Dependency Matrix** and then **Create Blank Matrix**. Type a name and select the package where you want to save it in the project and click **OK**.
- From the package shortcut menu in the Browser, select **New Diagram** and then **Dependency Matrix**.
- In the custom diagrams toolbar, click the **Dependency Matrix** button. Type a name and select the package where you want to save it in the project and click **OK**.

To change the dependency matrix properties

1. In the **Dependency Matrix Specification** dialog box (see “The Dependency Matrix Specification dialog box, tags group” on page 599), open the **Tags** group.
2. In the <<matrix template>> subgroup, select balded tag, marked with icon  from the list and edit its value in the **Value** field.

Dependency Matrix View

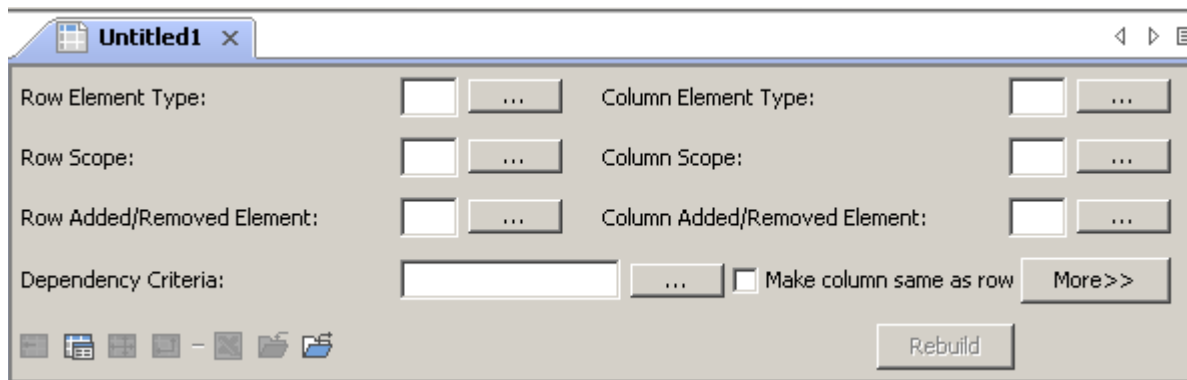









Figure 288 -- Dependency Matrix View

Row Element Type	Click the “...” button to select the element or multiple element types to show in rows of the dependency matrix.
-------------------------	--

Column Element Type	Click the “...” button to select the element or multiple element types to show in columns of the dependency matrix.
Row Scope	Click the “...” button to define the scope of the model (packages/profiles), from which elements should be displayed in rows of the dependency matrix.
Column Scope	Click the “...” button to define the scope of the model (packages/profiles), from which elements should be displayed in columns of the dependency matrix.
Row Added/ Removed Element	<p>+<owner>::<element> value shows the row that is added to the matrix. -<owner>::<element> value shows the row that is removed from the matrix.</p> <p>Click “...” button to add or remove row from the dependency matrix table. The Add/Remove Elements dialog box opens. For more information about this dialog box, see “The Add / Remove Elements dialog box” on page 598.</p>
Column Added/ Removed Element	<p>+<owner>::<element> value shows the column that is added to the matrix. -<owner>::<element> value shows the column that is removed from the matrix.</p> <p>Click “...” button to add or remove column from the dependency matrix table. The Add/Remove Elements dialog box opens. For more information about this dialog box, see “The Add / Remove Elements dialog box” on page 598.</p>
Dependency Criteria	<p>Click the “...” button to define what relations between row and column elements to display in the matrix cells.</p> <ul style="list-style-type: none"> • UML Relations. The matrix cells will display the existence of relationships between row elements and column elements. • Tagged Value. The matrix cells will display relations through tags if matrix row elements have tagged values pointing to matrix column elements or vice versa. • Properties. The matrix rows will display relations through properties if matrix row elements have properties pointing to column elements or vice versa.
Make column same as row	If selected, sets the column filter setting to the same as row.
More>>	Click More>> button to expand the toolbar and find advanced matrix filters areas.

Row Property/ Column Property	<p>Select item from the combo box to select row/column elements, filtered by some property. Only entries that meet selected property values will be displayed in the rows/columns. Possible choices:</p> <ul style="list-style-type: none"> • Applied stereotype; • Visibility; • Tagged Value; • To Do.
Row Property Value Column Property Value	<p>The “...” button is enabled only when some Property is selected. The corresponding dialog box with available property values opens. Multiple property value selections are available.</p>
Rebuild	<p>Columns and rows rebuild action, which rebuilds element lists according to the filters configuration.</p>
Add Remove Elements	<p>Add/remove elements and diagrams from/to Matrix rows or columns independently from the filter settings.</p> <p>Click the  button to open the Add Remove Elements dialog box. For more information about this dialog box, see “The Add / Remove Elements dialog box” on page 598.</p>
Matrix Properties	<p>Click the  button to open the Dependency Matrix Specification dialog box. For more information about this dialog box, see “The Dependency Matrix Specification dialog box, tags group” on page 599</p>
Quick Diagram Layout	<p>Click the  button to automatically layout the matrix cells to default width and height.</p>
Change Axes	<p>Click the  button. Columns and rows will be exchanged with each other.</p>

Save as *.csv	<p>Click the  button. The Save dialog box opens. Select location for a file and enter a file name (default name will be set the same as Matrix Name). The file will be saved in Comma Separated Values (*.csv) format. The file can be opened with MS Excel, Open Office, MC Excel imported into databases.</p>
Save Configuration As Template	<p>Click the  button to save the filters configuration as a matrix template. The saved template can be imported into the Dependency Matrix Template dialog box and used in other projects.</p>
Load Matrix Template	<p>Click the  button to open the Load Matrix Template dialog box. Select a template from the list and click OK.</p>

Dependency Matrix pane

After the dependency criteria (filter options) are specified, click the **Build** button. The Dependency Matrix pane will be opened.

8 MODEL ANALYSIS

Dependency Matrix

Dependencies between elements are displayed in cells. Rows and columns display elements, which were specified in the Matrix View fields.

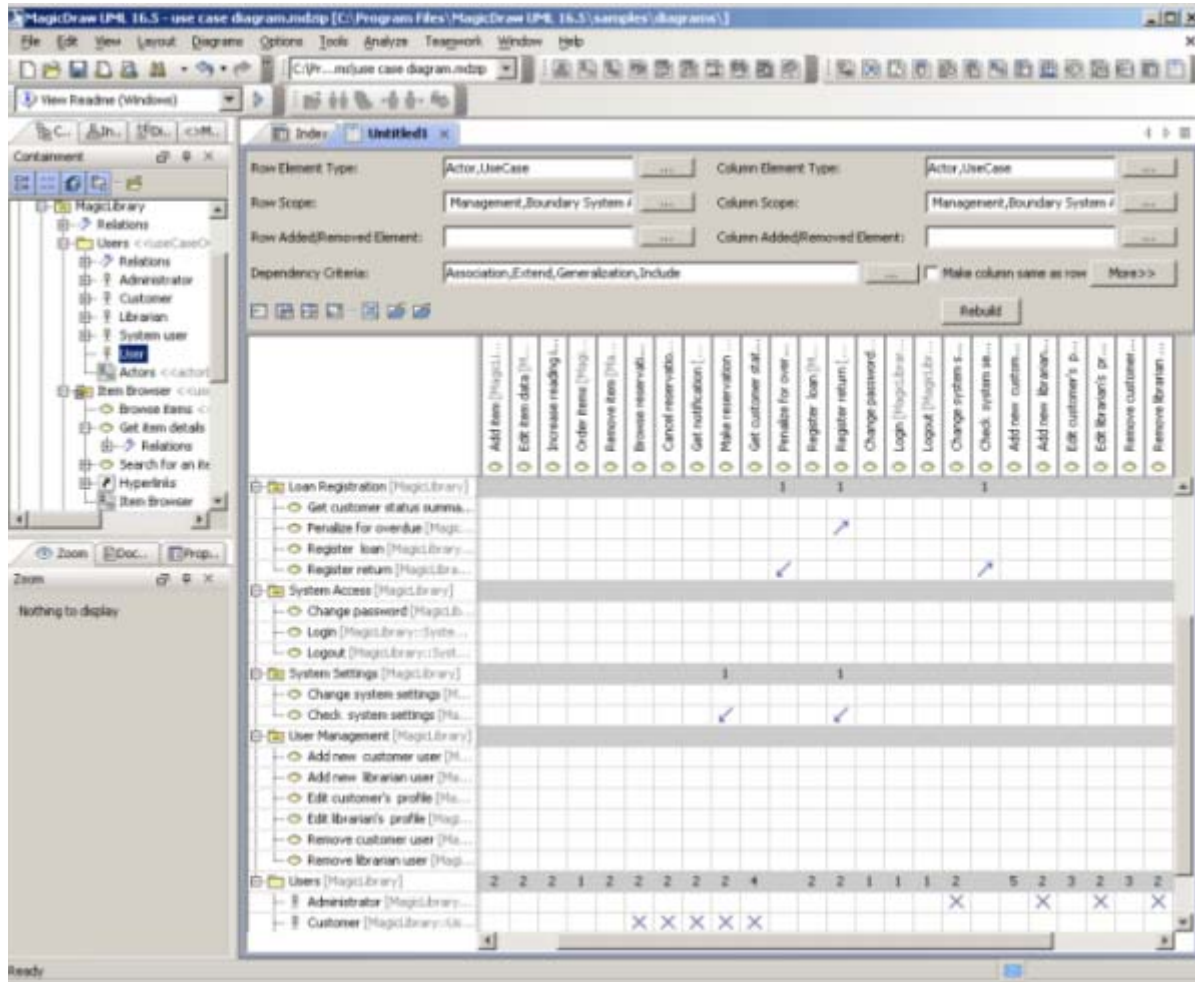


Figure 289 -- Dependency Matrix pane

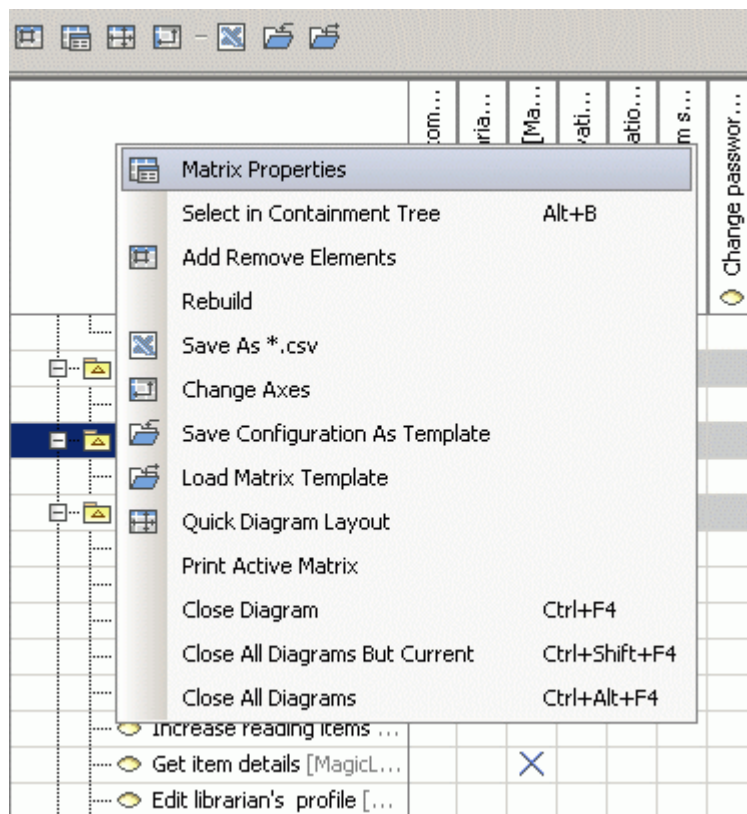
An Icon with dependency direction representation is displayed in a cell if the single dependency is presented in this cell. Arrows can be:

- One directional arrow - direction shows that element is dependent on the element, to which direction it points;

- Line - shows that elements depend on each other;
- "X" icon - appears when multiple dependencies are presented in the cell.

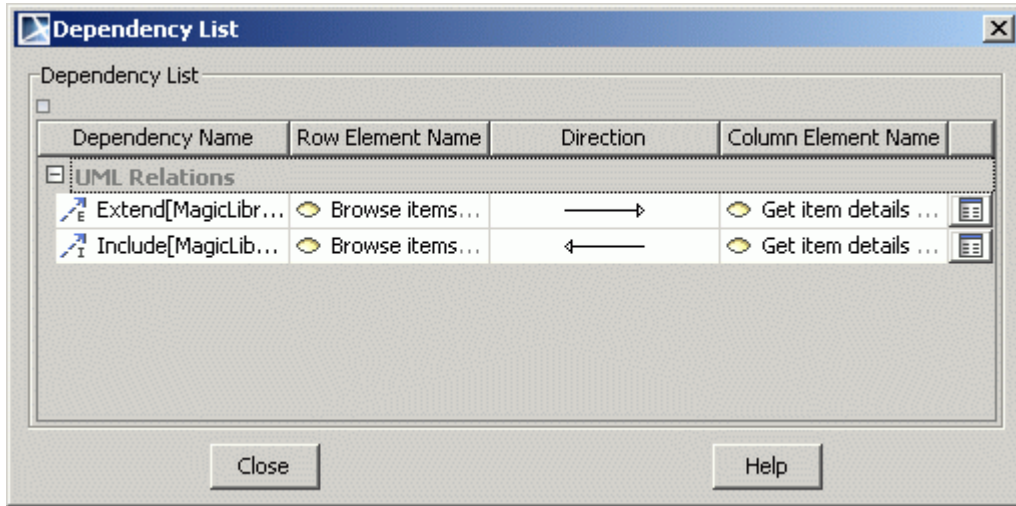
The number of dependencies between package elements is displayed in a cell where packages are intersecting with any element.


Right-click on the empty space in the Dependency Matrix pane to open the shortcut menu:



To open the relation specification dialog box of the dependent element

1. Double click a cell that is not empty or select it and from the shortcut menu, select **Dependency List**. The **Dependency List** dialog box opens.



2. Click the  button near the selected dependency to open the corresponding dependency Specification dialog box.

Working with a Dependency Matrix Template

Matrix properties and filter configurations are stored in MagicDraw. The matrix configuration is called the matrix template. The matrix template is used to save a matrix independently of a project and to share them with other users. The user can create a matrix based on a predefined template.

The matrix template can be imported and exported as a file. Filters and Matrix properties will be saved as a matrix template.


To create a Dependency Matrix from a Template

1. From the **Analyze** main menu, select **Dependency Matrix** and then **Matrix Templates**. The **Dependency Matrix Templates** dialog box opens.
2. Select the template from the list and click **Create Matrix**, or click **OK**. After closing the dialog box, click **Rebuild** in the Matrix View.


To load a Dependency Matrix Template

1. In the Matrix View, click the **Load Matrix Template** button . The **Load Matrix Template** dialog box opens.
2. Select a template from the list and click **OK**. The dialog box is closed and template values appear in the rows/columns filter fields in the Matrix View.
3. Click **Rebuild** in order to apply the template filter for the dependency matrix.


To save a configuration as a Dependency Matrix Template

1. Specify the desired filter values for rows and columns.
2. In the Matrix View, click the **Save Configuration As Template** button . The **Save** dialog box opens.
3. Type a name for the template, select a location and click **Save**.

To export a Dependency Matrix Template

1. In the **Dependency Matrix Templates** dialog box, click the **Export Selected Matrix template** button . The **Save** dialog box opens.
2. Type a name for the template, select a location and click **Save**.

To import a Dependency Matrix Template

1. In the **Dependency Matrix Templates** dialog box, click the **Import New Matrix template** button . The **Open** dialog box opens.
2. Select a matrix template and click **Open**. The template with defined properties will be added to the templates list.

Dialog boxes in Dependency Matrix functionality

Dependency Matrix Templates dialog box

From the **Analyze** main menu, select **Dependency Matrix** and then **Matrix Templates**. The **Dependency Matrix Templates** dialog box opens.

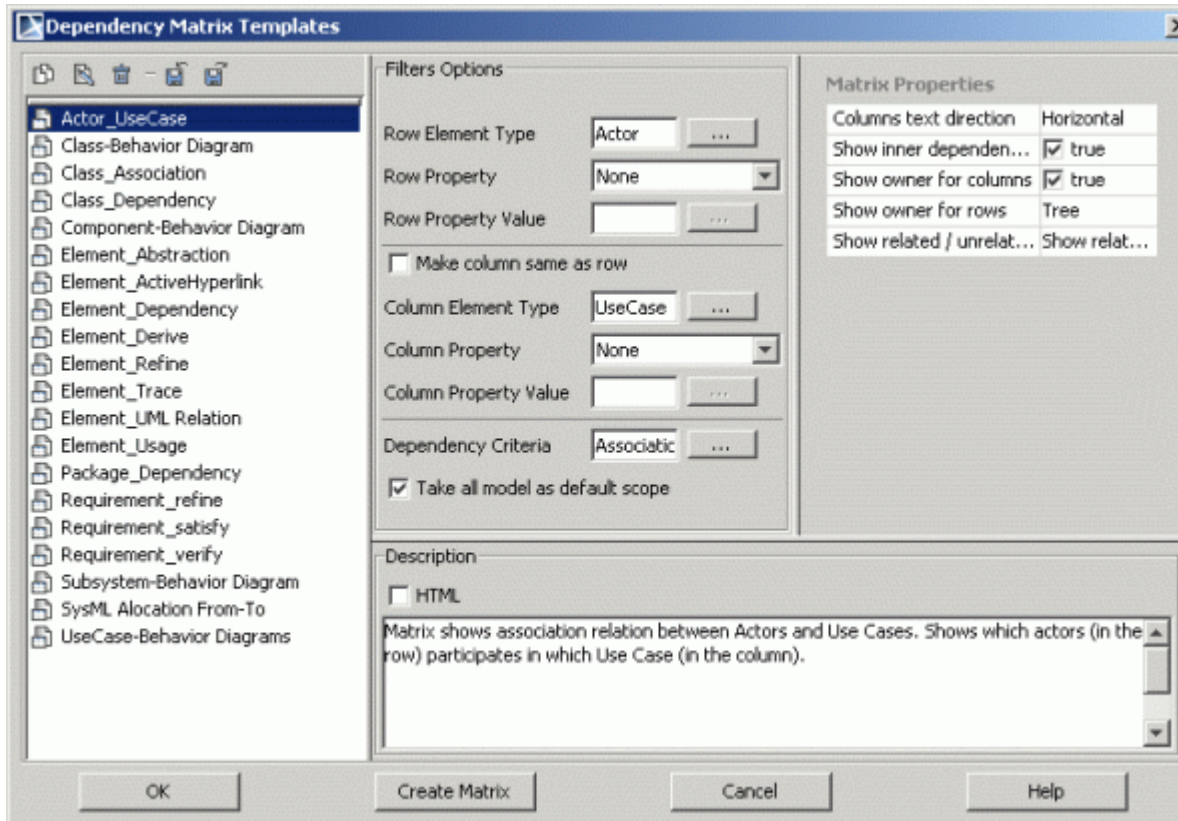







Figure 290 -- The Package Dependencies dialog box

Box name	Function
Clone Selected Matrix template	Select the template and click the  button. A copy of the existing matrix template will be created with the same properties and filter options.
Rename Selected Matrix template	Select the template and click the  button. The template name will be set to editable mode for renaming.
Remove Selected Matrix template	Select the template and click the  button. The template will be removed from the templates list.
Import New Matrix template	Click the  button to import a template. The Open dialog box opens.
Export Selected Matrix template	Click the  button to export a template. The Save dialog box opens.
Filters Options	For more information, see “Dependency Matrix View” on page 587.
Take all model as default scope	If selected, upon matrix creation, the whole model will be taken as the scope (later scope can be changed in Dependency Matrix View). If cleared, matrix will be empty until specifying scope.
Matrix Properties	

Columns text direction	Changes text direction in column headers. Possible choices: <ul style="list-style-type: none"> • Horizontal • Vertical
Show inner dependencies	If selected, the number of dependencies among container elements will be shown in the cell where the container is intersecting with other elements.
Show owner for columns	If selected, displays the element owners in the column headers.
Show owner for rows	Displays the element owner in the row headers. Possible choices: <ul style="list-style-type: none"> • Full qualified name • Hide • Tree
Show related/unrelated elements	Allows showing elements with or without their dependency criteria. Possible choices: <ul style="list-style-type: none"> • Show all; • Show related elements only; • Show unrelated elements only.
Description	Displays the matrix template description. Select the HTML check box to edit text using the advanced style.
Change Exports	The Export Module dialog box opens. You can change the package set, selected for export.
OK	Closes the dialog box and adds changes to the Matrix View fields according to the selected template.
Create Matrix	Closes the dialog box and creates the dependency matrix, applying filters and options from the selected template.
Cancel	Exits the dialog box without saving changes.
Help	Displays MagicDraw Help.

The Add / Remove Elements dialog box

In the Dependency Matrix View, click the  button to open the **Add Remove Elements** dialog box.

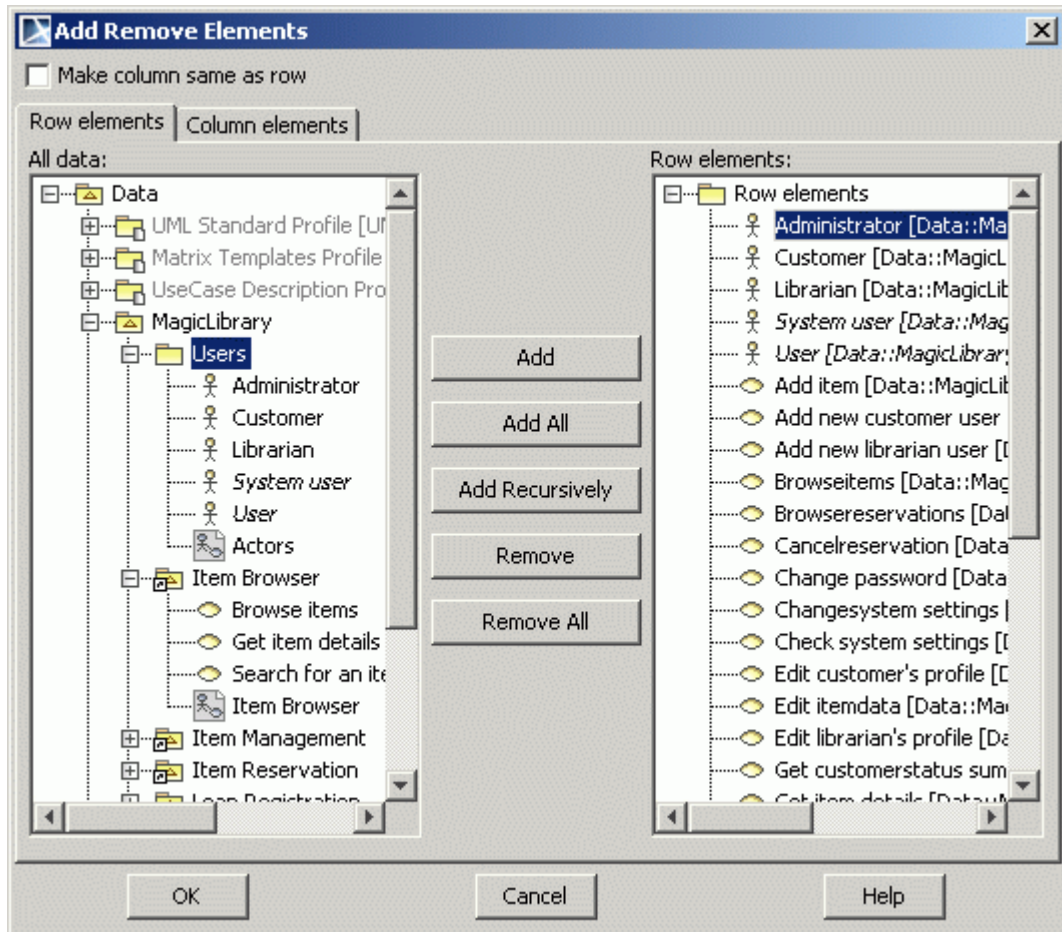


Figure 291 -- The Add / Remove Elements dialog box

Box name	Function
Make column same as row	If selected, applies the same element combination that was set for row, to column.
Row elements	Displays a list of all model elements and chosen elements to display in the matrix rows.
Column elements	Displays a list of all model elements and chosen elements to display in the matrix columns.
Add	Adds the selected element from the All data list to Row (Column) elements list without adding its inner elements.
Add All	Adds all model elements from the All data list to Row (Column) elements list.
Add Recursively	Adds the selected element from the All data list to Row (Column) elements list together with its inner elements.
Remove	Removes the selected element from the Row (Column)elements list.
Remove All	Removes all elements from the Row (Column)elements list.
OK	Saves changes and closes dialog box.
Cancel	Closes dialog box without saving changes.
Help	Displays the MagicDraw help.

The Dependency Matrix Specification dialog box, tags group

- In the Browser, select the matrix element and from the shortcut menu, choose **Specification**.

- From the Dependency Matrix pane shortcut menu, click **Matrix Properties**.

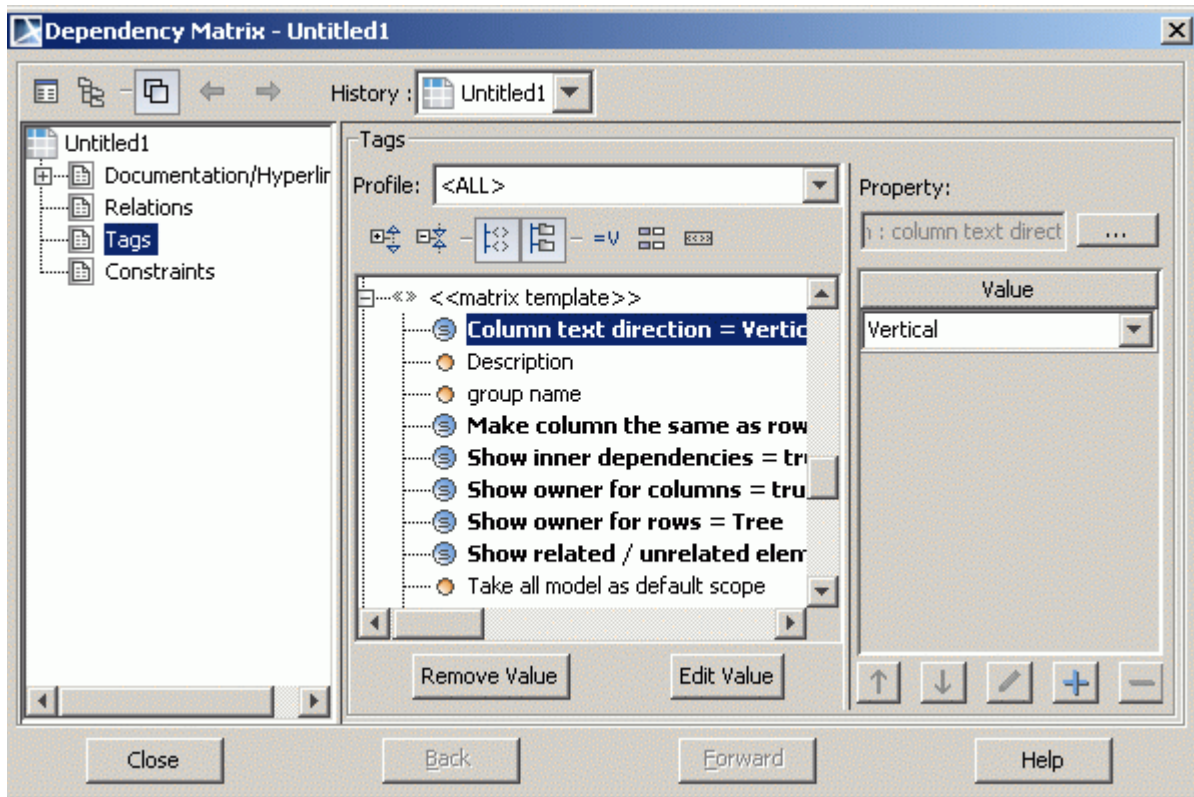


Figure 292 -- The Dependency Matrix specification dialog box, Tags group

This dialog box can be used to change matrix property values. In the `<<matrix template>>` tags group, select bolded tags and in the right open window, change the default value to the desired one.

Validation

[View Online Demo](#) Validation

NOTE This functionality is available in Architect and Enterprise editions only.

Introduction

MagicDraw has the functionality to check the created models. It consists of:

- A set of validation rules. Each validation rule captures some imperative conditions, which must be checked against the model. Validation rules are specified as invariant constraints in the model.
- One or more validation suites (modeled as packages). A validation suite is a simple concept of grouping the validation rules into meaningful groups, so that the collection of rules can be applied.

To run the validation, select some suites and validation scope - either the entire model or some part of it. When the validation is run, each rule from the suite is evaluated for each suitable element in the validation scope. Each element that violates the rule (constraint evaluates to false) is reported in the results table.

Since rules and suites are model elements, they can be manipulated using the standard MagicDraw modeling means - they can be copied, moved, and edited in the model; they can be refactored into modules, to facilitate reuse in other projects, placed in the Teamwork Server for exchange, etc. And of course this approach allows editing predefined rules and defining new, custom rules for models and profiles.

Constraint Types

Each validation rule, modeled as a constraint has a target classifier property. This property determines on what type of element this rule applies. Thus the usual level - metalevel separation appears. Constraints that are defined on some particular classifiers are evaluated on the instances of these particular classifiers when validating. Inheritance is taken into account - instances of the subclasses of the class are also validated.

Thus there are 3 types of constraints that MagicDraw can evaluate:

- **Classifier level constraints.** Constraints that are placed on the classes, datatypes and other classifiers of the model are evaluated on all the instances of these classifiers - i.e. those `InstanceSpecifications` that have the particular classifier set as their type.
- **Constraints on metaclasses.** When a constraint is placed on a metaclass (one of the classes in the *UML Standard Profile::UML2 Metamodel*), this constraint is evaluated on all the model elements of that kind. E.g. if the constraint is placed on Actor metaclass, then this constraint applies to all the actor elements in the model. The following is an example of rule (specified in OCL2.0), which mandates that all actor names in the model must be capitalized:

context Actor inv capitalize:

```
let startswith:String = name.substring(1,1) in
```

```
startswith.toUpperCase() = startswith
```

These constraints are useful for specifying generic rules, which must apply on all the model elements of particular kind.

- **Constraints on stereotypes.** When a constraint is placed on some stereotypes of the profile, that constraint applies to all the model elements that have these stereotypes applied to them. These constraints are useful when creating domain specific profiles. When adapting UML to some specific modeling domain, a profile is usually created with extensions for that domain - stereotypes, tags etc. The constraints on these stereotypes allow enforcing the rules of that domain.

It is advisable not to mix the constraints from different metalevels into one suite (constraints on classifiers versus constraints on stereotypes and metaclasses).

Predefined Validation Suites

There are several validation suites (collections of validation rules) predefined in the profiles that come with MagicDraw. Since validation rules and validation suites are concepts, stored in the model, the availability of list of validation suites for validating depends on what profiles the model includes.

UML Standard Profile brings two predefined suites with it. These two suites are present in all models:

- UML completeness constraints;
- UML correctness constraints.

Completeness suite has a collection of rules, which check if a model is complete, that there are no gaps, and the essential information fields in the elements have been filled in (e.g. checks that all the properties have type specified etc.).

Correctness suite has a collection of rules, which check common mistakes while modeling in UML2 (NOTE: this collection is not exhaustive).

Additionally, there are validation suites for each of these modeling domains - XML schemas, DDL, Java, C++ plus DoDAF and SysML, if any. These validation suites are defined in the corresponding profiles of these modeling domains, hence they are included automatically when you start modeling in that domain.

For example - if you create a new XML schema diagram, XML schema profile will be automatically included in your model and this profile brings in XML schema validation suite with it. So, from that moment, XML schema validation suite is available in the project.

Validating

To run the validation, you have to select a group of rules to be validated (validation suite) and indicate which part of the model to check (validation scope - either the entire model or some part of it).

To validate UML model for correctness

1. Open model. From the **Analyze** main menu, select **Validation** and then **Validation** again. The **Validation** dialog box appears.

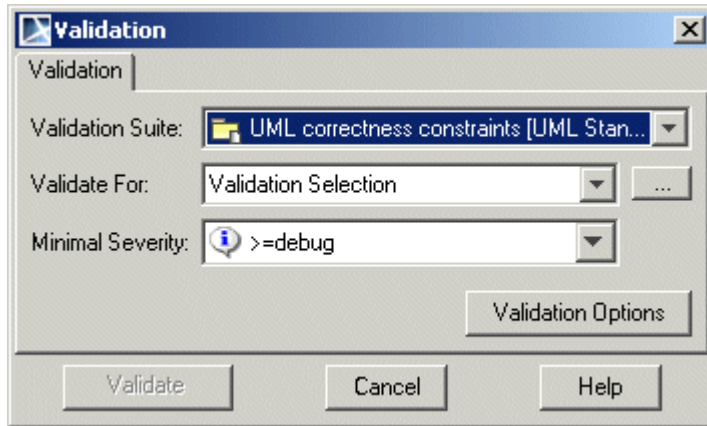


Figure 293 -- Validation dialog box.

2. In the **Validation Suite** combo box, select the UML correctness constraints . All available validation suites are listed here.

The list of available validation suites depends on the open project - the validation suites and validation rules are stored in the model as normal model elements. By default, a project has two suites - UML completeness constraints and UML correctness constraints - defined in the Standard profile.

If a project uses other profiles/modules - such as Java/XML schema/DDI profile, these profiles bring in their own predefined suites. And you can create your own validation rules and group them into a suite and this suite will be available in this combobox.

3. Select validation for Validation Selection (if you want to run validation on the entire model select **Whole Project** here) then click the “...” button to select the scope. As usual a dialog box for element selection opens:

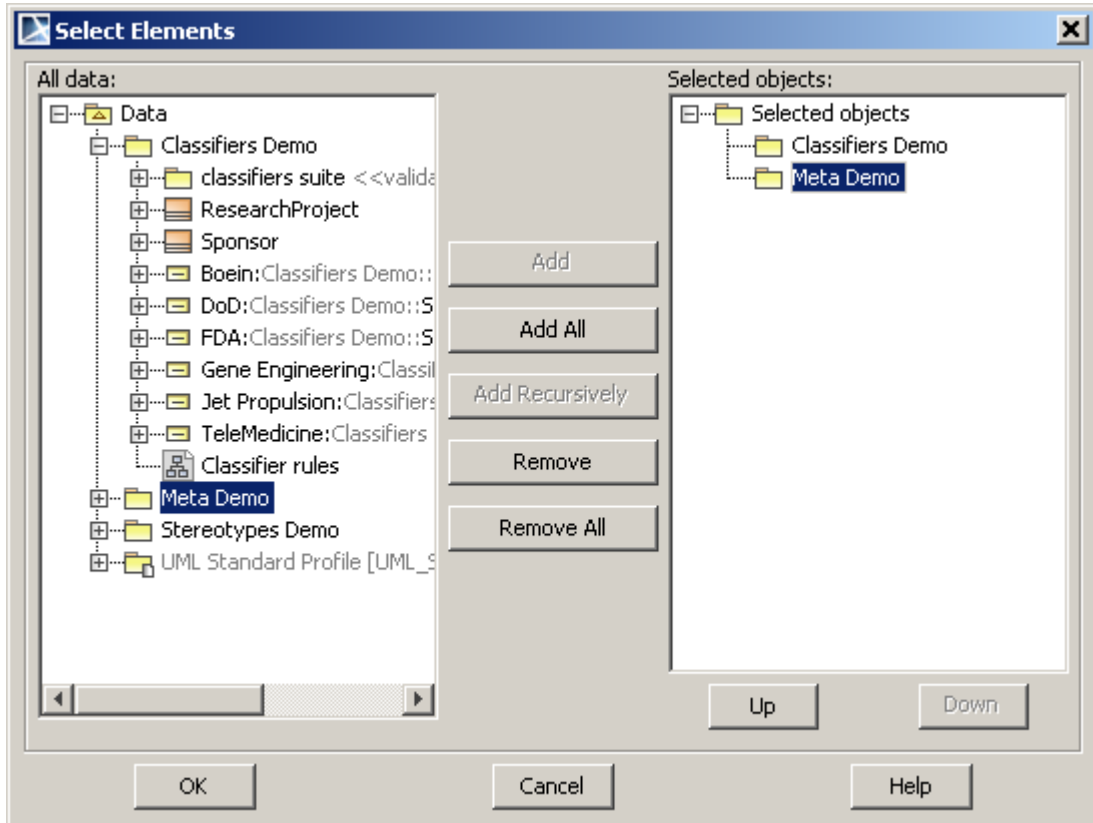


Figure 294 -- Select Elements dialog box

4. In the **Select Elements** dialog box, select packages and/or elements for validation.
NOTE: validation is always recursive, hence if you select a package for validation, you do not need to select its inner elements (no need for Add Recursively button). In the case depicted above, all model elements in the *Classifiers Demo* and *Meta Demo* packages will be validated. Adding *Data* package is equivalent to validating the entire model.
5. Select the **Minimal Severity** level. By default, debug is chosen as minimum severity level. Since debug is the lowest possible severity level, all validation rules will be run.
6. Click **Validate**. Validation results are displayed in the **Validation Results** window.

Validation Results Window

Validation results are displayed in the Validation results window.

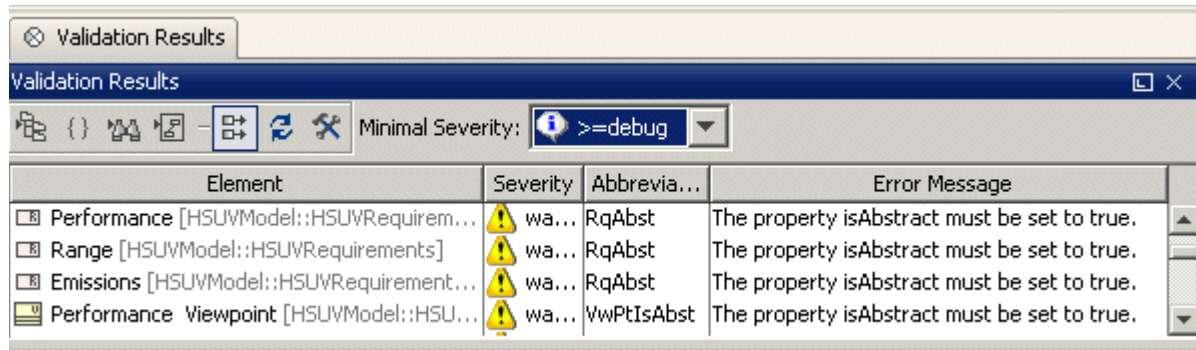


Figure 295 -- Validations Results view.

Window opens automatically after the validation has ended. The Validation result window has the following columns:

Column	Description
Element	Elements, which violate the constraint rule are shown here.
Severity	Rule severity violations.
Abbreviation	Simple short strings showing the abbreviation of the violated constraint. Mostly used for sorting/grouping.
Error Message	Error texts of the violated constraints.

If the validation rule is incorrect, **Validations Results** view will show the reason. This rule is then excluded from checking models.

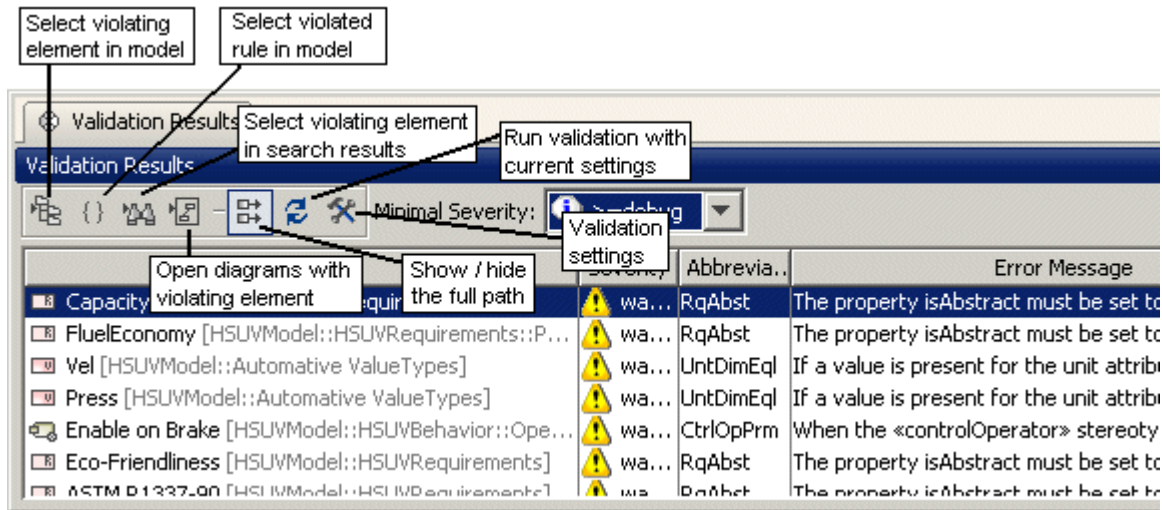



Figure 296 -- Buttons of Validation Results view.

To validate a model

We will validate SysML model - SysML.mdzip for correctness and completeness. The model is located in MagicDraw installation directory / samples. This model requires that SysML plugin would be installed. This can be done from **Help** menu **Resource / Plugin Manager**.

1. Open model. From the **Analyze** menu, select **Validation** command, and then **Validation**. The **Validation** dialog box opens.
2. Select **SysML Validation Constraints suite** and click **Validate**. The **Validations Results** view opens.
3. Narrow the validation scope. Click the **Run validation with a new**


settings  button, the **Validation** dialog box opens. Change the scope to validation selection and select **SI Value Types** package, click **Validate**. The **Validations Results** view is refreshed with new options.

4. Double click the violating element *kg* in the results view or click **Select in**

Containment Tree



button. The element is selected in the model.

5. Fix the problem and click the **Refresh**  button to rerun the validation suite with the same options and refresh validation results. Refreshed results do not include element kg.
6. Select other violating element N in the results view and click **Open all**

Diagrams Containing the Selected Elements



button to open all the diagrams, containing the selected model elements.

When the validation result view is opened and diagrams are shown on the screen, elements and links of the diagram, which have at least one result in the validation result view, are highlighted:

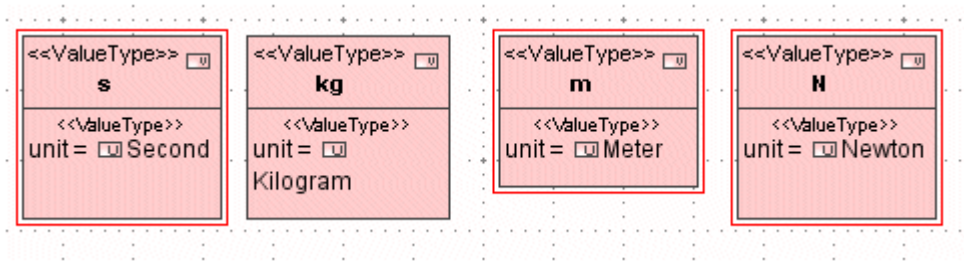


Figure 297 -- "N" element violating validation rule is highlighted.

In general, highlighting depends on the validation rule severity:

- Error - red
- Warning - yellow
- Debug - gray

- Info - black

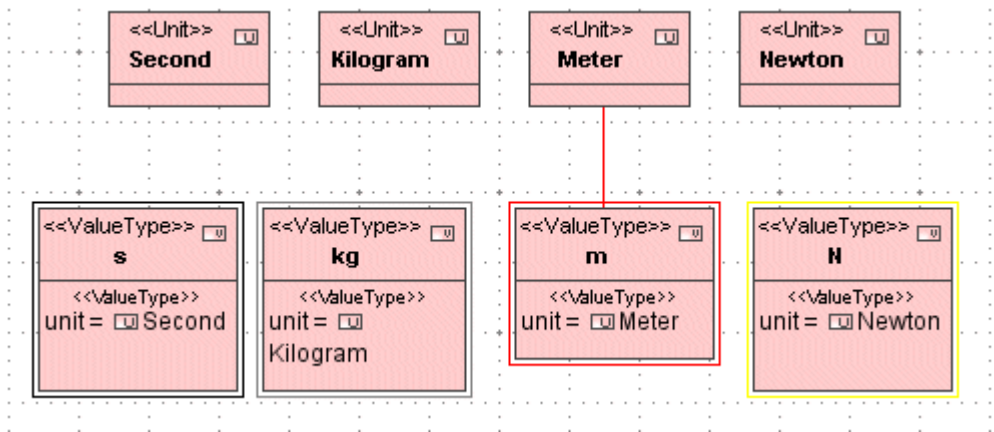






Figure 298 -- Elements and links violating constraints are highlighted.

If the element violates multiple rules, the color of the most severe rule is used.

Highlighting is shown until the rule is violated or the Validations Results view is not closed.

Other features available for the Validations Results view:

- By clicking the **Select Rule in Containment Tree**  button, you may select rule in the model.
- You may move the selected results of the validation to the search result window by clicking the **Move to Search Results**  button.
- The **Open all Diagrams Containing the Selected Elements**  button opens all the diagrams, containing the selected model element.
- The Show / hide the Full Path names  button shows element location in the model path.

Validation Rules

The validation rules are modeled as UML2 constraints. This approach allows treating the validation rules as simple model elements. They can be handled using usual modeling mechanisms. They can be copied, moved around in the model, refactored into a separate module, stored in the teamwork server for easy information exchange, etc.

Since constraints can have different semantic meanings in UML2, a special type of the constraint – invariant constraint is used for modeling validation rules.

To distinguish these constraints from the other types of constraints, `<<invariant>>` stereotype should be applied to them.

Additionally, validation rules require other pieces of information – severity level (for sorting/filtering), abbreviation string (a short string, for easy recognition) and error message (complete description of error explanation). This information is displayed in the validation result view.

For storing this information, a special stereotype `<<validationRule>>`, derived from the `<<invariant>>`, is used. If you want to run this constraint as validation rule, use the former stereotype. If you have created the constraint just for documentation purposes and do not intend to run it, the latter constraint is sufficient.

Validation rules can be placed anywhere in the model (where UML2 constraint can be placed), however, usually they are stored in the classifier, which is constrained – classes, datatypes, etc. (for classifier level constraints), stereotypes (for meta-classifier level constraints). This convention breaks down for the constraints, placed on metaclasses (since these classes are stored in read only profile). In this case, place constraints wherever you like (e.g. group them into a package).

To create a validation rule on a metaclass

1. Select a package where you want to place the rule.
2. Right-click this package, select *New Element*, and then *Constraint*.

To create a validation rule on a classifier or stereotype

1. Open their specification, select *Constraints* section, and click **Create**.

Example:

Let's say we have 2 stereotypes - `<<product>>` and `<<part>>`. We want to place a validation rule, that products must have at least one part in them.

1. Open the product stereotype specification, select *Constraints* section, and click **Create**.

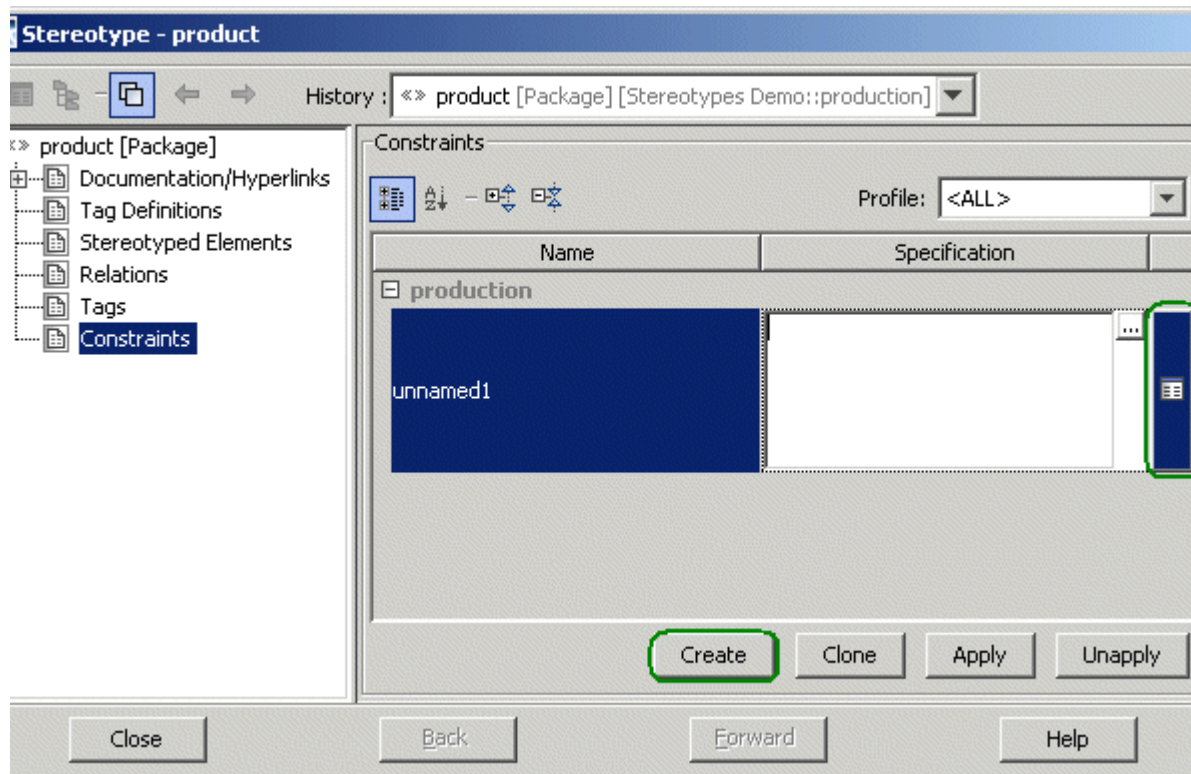


Figure 299 -- Creating constraints

2. The name of the constraint and the expression can be specified right away, but since we want to specify more information, we need to open the specification of a newly created constraint (press the button on the right of the constraint).

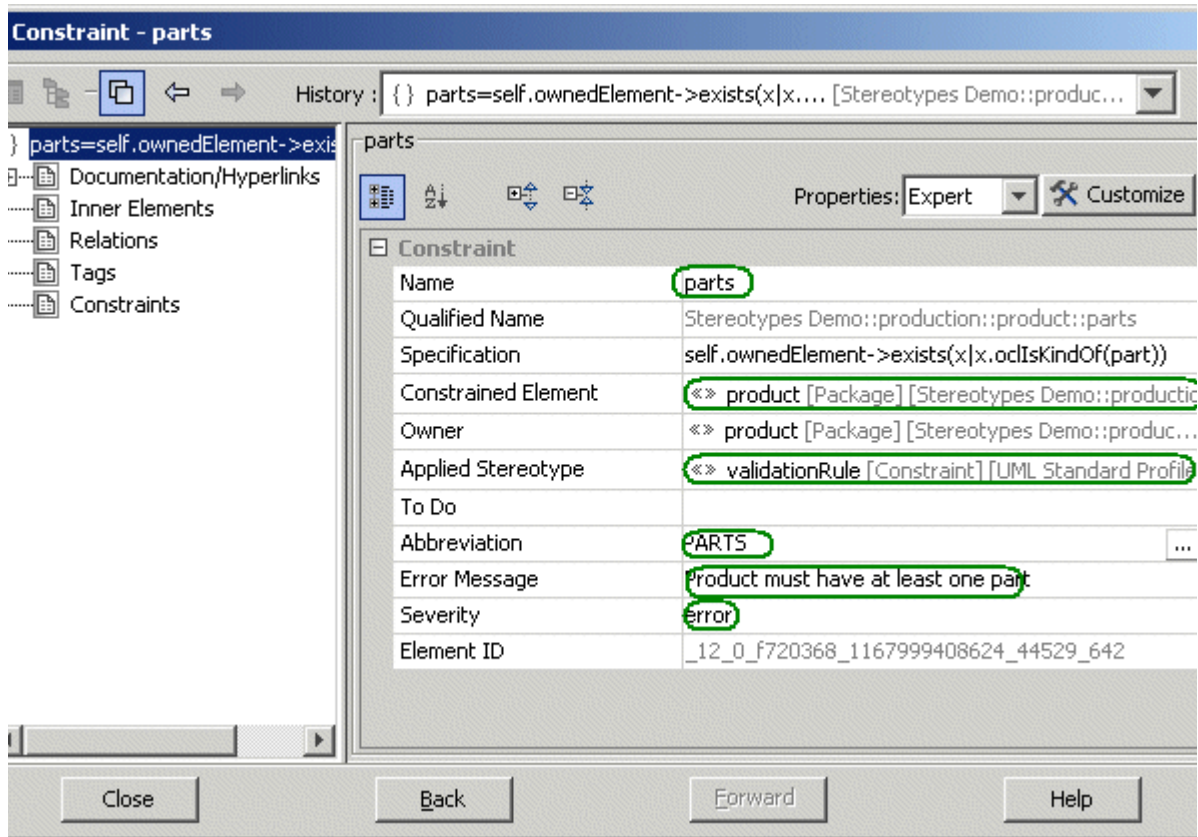


Figure 300 -- Specifying details of the constraint

3. In the specification panel of the constraint, specify the constraint name. Then ensure that the *Constrained Element* field points to the necessary classifier (*product* stereotype in our case). If we have created the constraint as described here, this field will be filled automatically. If we have created the constraint through the right-click, *Create Element, Constraint* route (e.g. constraint for metaclasses), we will need to specify the constrained element manually. For constraints on metaclasses select the appropriate element from the *UML Standard Profile::UML2 Metamodel*. In UML2 the constrained element field is multivalued, but only single value is supported for validation rules.

4. Now apply the <<*validationRule*>> stereotype on this constraint. Additional fields will open on the pane (*Abbreviation*, *Error Message* and *Severity*). If these fields do not open automatically, click **Customize** and then **Reset to Defaults** in the open customization dialog box (you can also access these fields in the Tags section of the specification).
5. Fill in the values for those fields.

Usage of the *Severity* levels (approximate guidelines):

- **DEBUG.** This severity level should be assigned only to those validation rules, which fit the description of INFO, but are too numerous and annoying to constantly bother the user.
- **INFO.** Situations, which might be interesting to the user.
- **WARNING.** Used for less severe situations than ERROR, which are not errors per se, but have a high probability of causing errors. A good example would be – In Java model user redefines equals() method of the class, but does not redefine hashCode(). This is a dangerous coding situation.
- **ERROR.** Normal error message. For ordinary, run-of-the-mill errors.
- **FATAL.** Used for the errors, which lead to model corruption or are not valid from the UML metamodel structure viewpoint. There should be few or no validation rules of this level since MagicDraw automatically precludes such situations. This level is mostly reserved for future use.

Abbreviation is a simple (and preferably short) string, for quickly distinguishing the validation rules among other rules and sorting. Acronyms and short forms, used in the domain of this validation rule can be used here (e.g. NPE for hypothetical NullPointerException check).

Error Message is a longer string, fully describing the invalid situation.

Now that we have all the peripheral information about the validation rule, let's specify the actual validation rule expression. Validating expression is stored in the *Specification* field. UML2 expression has 2 fields – *Language* and *Body*.

MagicDraw supports 2 languages for expressions, that can be evaluated:

- **OCL2.0** is used for validation rules, specified in OCL language (version 2.0 of the spec - 06-05-01 specification document from OMG).

- **Binary** is used for more advanced expressions, which are not easily expressed in OCL.

These expressions are written in Java, compiled, specified in the MagicDraw classpath. Then these expressions can be specified as validation rule expressions

Other languages are not evaluable (OCL1.5, English and others). They can be used for documentation purposes.

OCL Constraints

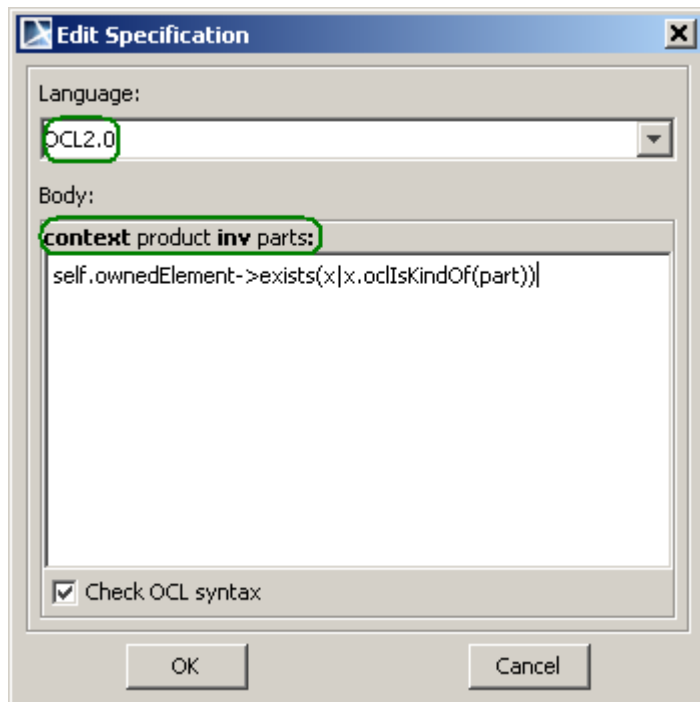


Figure 301 -- validation rule in OCL

Continuing our example, in the constraint **Specification** dialog box, click the button near the *Specification* field and open the **Edit Specification** dialog box. Select **OCL2.0** language. Observe that MagicDraw has automatically generated the header of the expression from the constraint information,

and we only need to specify the body of expression. The expression header is generated according to the following rules:

context <constrained element> <constraint type> <constraint name if any>:

Constraint type is one of the types, defined in the OCL2.0 spec:

- **inv** - when the expression is placed in the constraint with `<<invariant>>` stereotype applied.
- **def** – when the expression is placed in the constraint with `<<definition>>` stereotype applied.
- **init, derive** when the expression is placed in the default value of the property.
- **pre, post, body** when the expression is placed in the appropriate fields of operation.

Since our constraint is stereotyped with `<<validationRule>>` stereotype (derived from *invariant* stereotype), **inv** is shown in the header. Only invariant stereotype is used for the validation rules and are executed (plus derive expressions, when referenced from invariants – see “Advanced Topics” on page 623), other types of constraints can be used for documentation purposes.

MagicDraw checks the syntax of expression as you type. However this syntax check is not enough to catch all the errors. When the validation rule is run, additional checks are performed (semantic checks – such as checks for the existence of appropriate properties, type checks, multiplicity checks, etc.) to ensure that the expression can be evaluated correctly (internally, MagicDraw generates Java code from the expressions and then compiles it for execution).

Binary Constraints

For more information about Binary Constraints, see MagicDraw OpenAPI UserGuide.pdf.

Validation Suites

In MagicDraw, you can create your own validation suites or use one of the predefined ones as: *UML completeness constraints* and *UML correctness constraints*.

The validation suite defines the set of validation rules, which will be applied when validating. The purpose of the validation suites is to group constraints without duplicating them.

To create a new validation suite

We need to check the Oracle model for correctness, but not all constraints in *Generic DDL constraints* suite are suitable for our Oracle model. We will create a new suite with a narrow constraint collection.

1. From the **Analyze** menu, choose **Validation** command, and then **Validation Options**. The **Validation Options** dialog box opens.
2. Click the **Create New Validation Suite** button and name it *Oracle specific constraints*.

3. Define the validation rules in the **Validation Rules** pane.

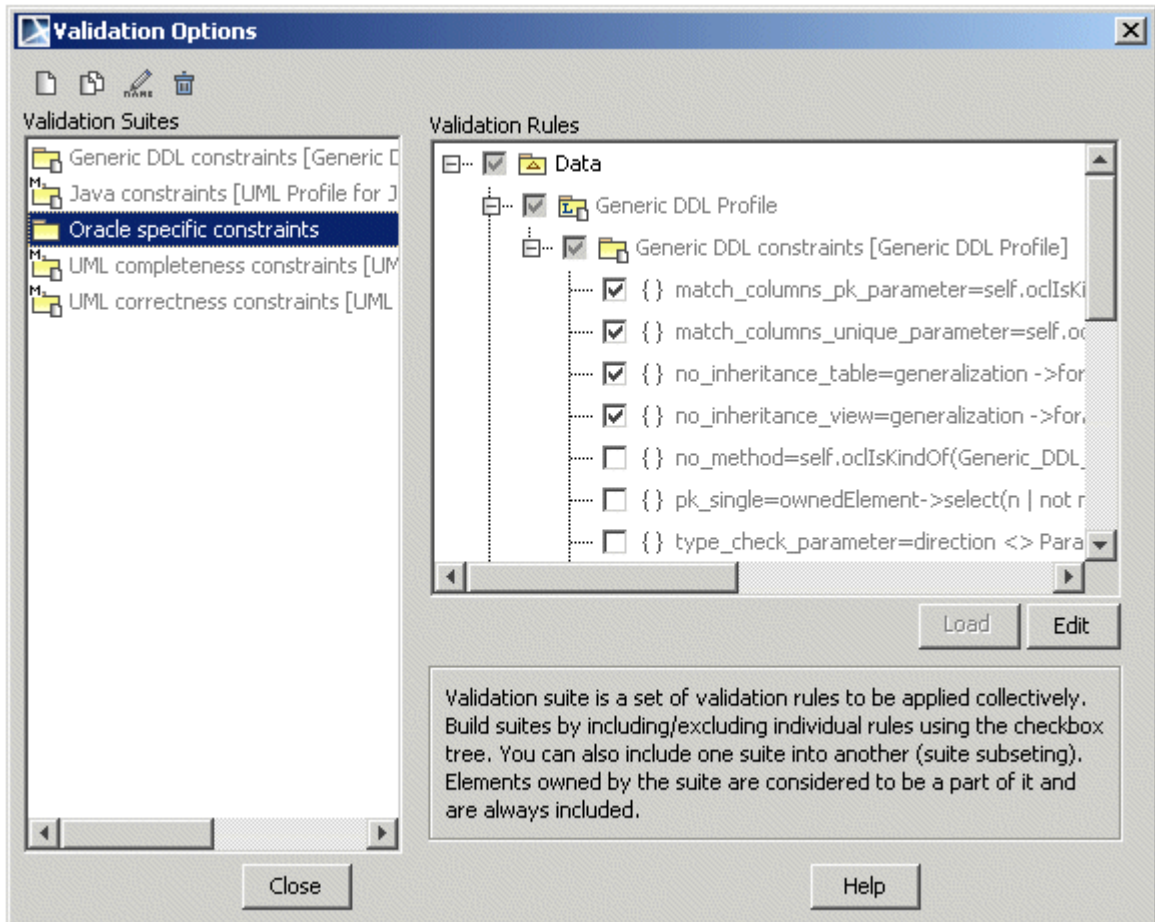


Figure 302 -- Parts of Generic DDL constraints suite are included into Oracle specific constraints suite.

The validation suite is stored in a model as a package, to which `<<ValidationSuite>>` stereotype is applied. The **Validation Suites** pane lists all such packages of the entire model as suites. So, the alternative method to create the validation suite is to apply `<<ValidationSuite>>` stereotype for a package.

When the user includes / excludes the constraint, the appropriate element import link is created/ deleted in the model.

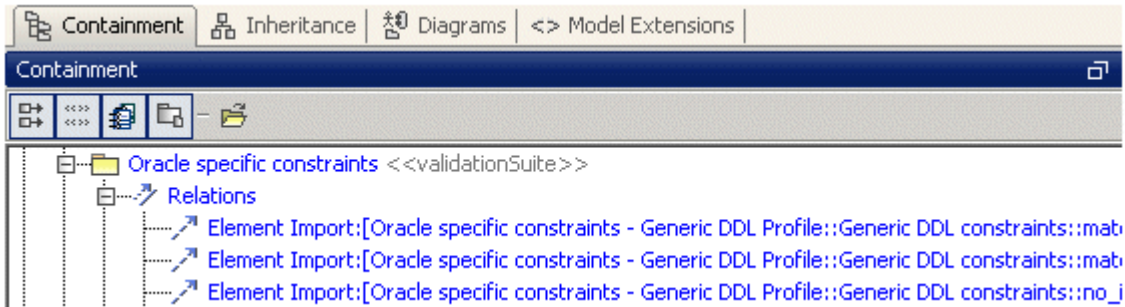


Figure 303 -- Element import relation showing in model that Oracle specific constraints suite includes other constraints.

Also, there can be constraints that are stored directly in the suite package - they are also considered as contained in this suite, and because they are physical in package those constraints can not be excluded from the suite through **Validation Options** dialog box. Typically, validation rules should be stored in a constrained element, but in cases when the constrained element is read-only, for example it is stored in a read-only profile, adding constraints to it requires profile editing and a separate constraints grouping is easier.

Constraint Tree

The constraint tree is shown in the right **Validation Options** pane. This tree shows all the constraints with `<<invariant>>` or `<<validationRule>>` stereotype applied, presented in the model, together with the appropriate grouping elements. Each item has a checkbox, indicating inclusion or exclusion of the constraint in the selected validation suite.

The constraint tree contains packages and other model elements. If it contains constraints, they are arranged according to their containment in a model. Additionally, this tree contains other validation suites. The user can include / exclude rules and these rules must suit the selected validation suite by selecting / unselecting these checkboxes in the tree.

To group two or more suites into one

We have created an abstract system model, and modeled its implementation with Java specific classes. To check this model completeness, correctness, and conformity to Java language by using three suites. We will combine all these suites to one in order to simplify the model checking.

1. From the **Analyze** menu, choose **Validation** command, and then **Validation Options**. The **Validation Options** dialog box appears.
2. Create a new suite, name it *General*, and select it.

3. Include the existing validation suites: *Java constraints*, *UML completeness constraints* and *UML correctness constraints* in the suite by selecting the checkbox in front of the packages in the **Validation Rules** panel.

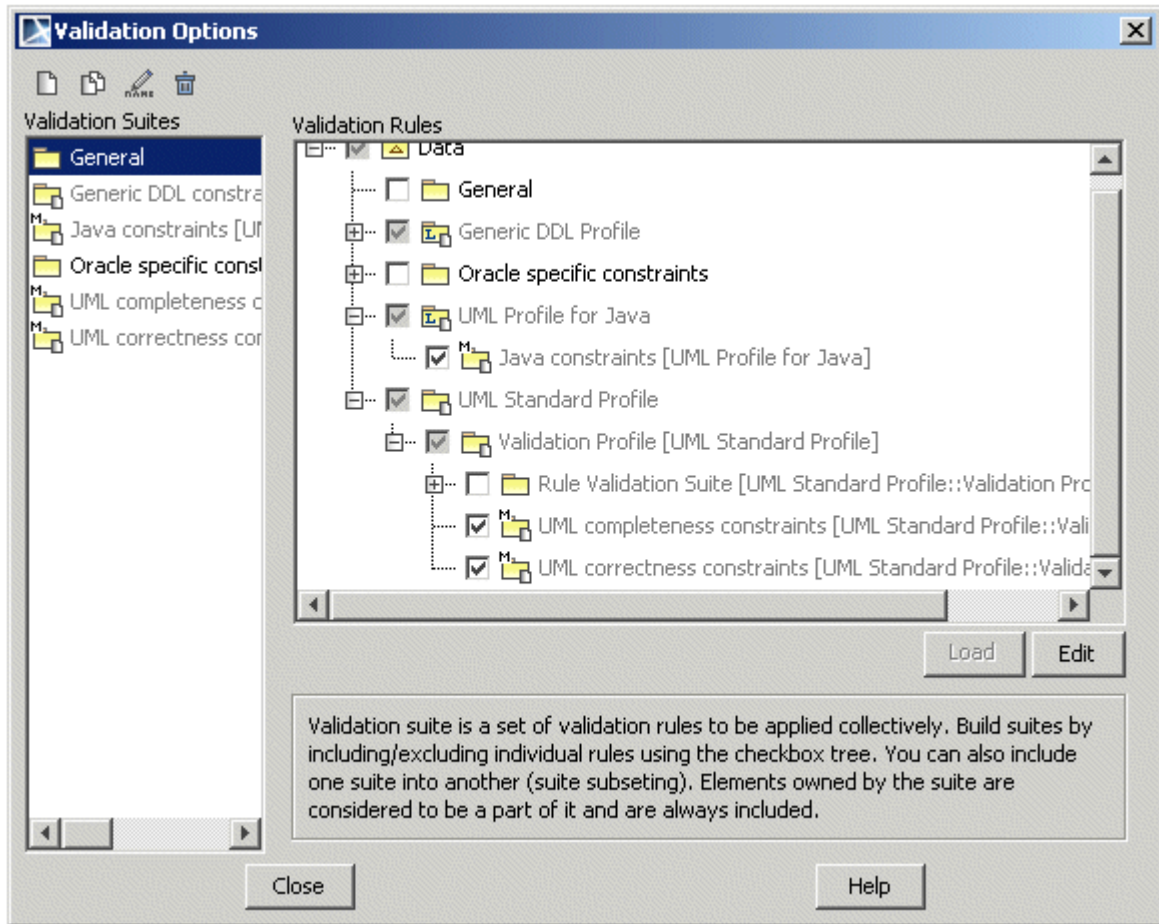


Figure 304 -- UML correctness, UML completeness, and Java validation suites included in the General

validation suite.

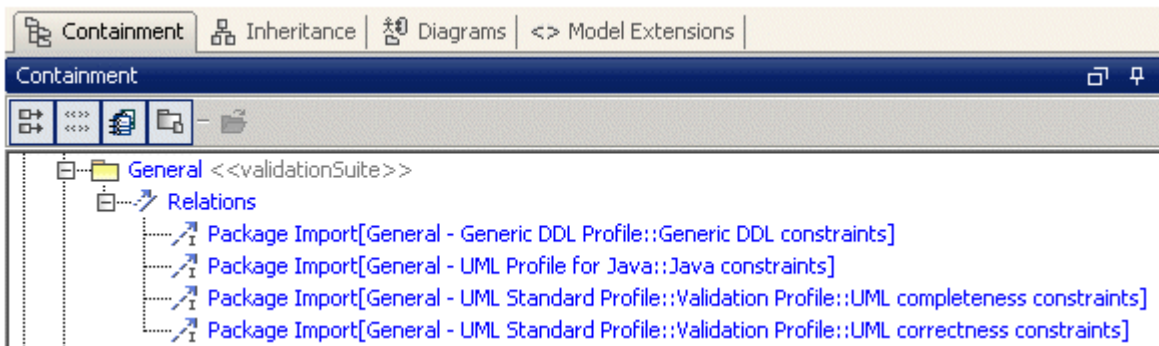


Figure 305 -- Package import is signifying in the model that one General suite includes the other as a subset.

To share constraints

Let's say we have created a validation suite with constraints and need to share it for other group members for their models validation. Validation rules/suites sharing is available through standard MagicDraw module mechanism. Package with constraints might be exported as module and used by any other project.

Another way of sharing constraints is copying them between projects. Since validation rules are simple model elements, any mechanism can be used on them.

1. From the model select *Java constraints* package to which <<ValidationSuite>> stereotype is applied.
2. From the package context menu select **Modules** and select **Export module**.
3. Save the exported package as *Java constraints.mdzip*.

Only constraints that are stored physically in the *Java constraints* package are exported together with the package.

Now the exported package can be used by other users and projects.

1. From the **File** menu, select **Use Module**. The **Use Module** dialog appears.
2. Select path to *Java constraints.mdzip* and select it to use.
3. Specify module import options.
4. Module is added into a project and the constraints can be used for validation.

The validation suite can be defined in the module, which is mounted as *read-only* in the project.

Read-only and **Autoload** module will not be loaded into the project, but will be visible through validations dialog box if <<ValidationSuite>> stereotype was applied to the exported package.

In this way the model and the profile will be smaller. And the validation suite is still visible through validation dialog box.

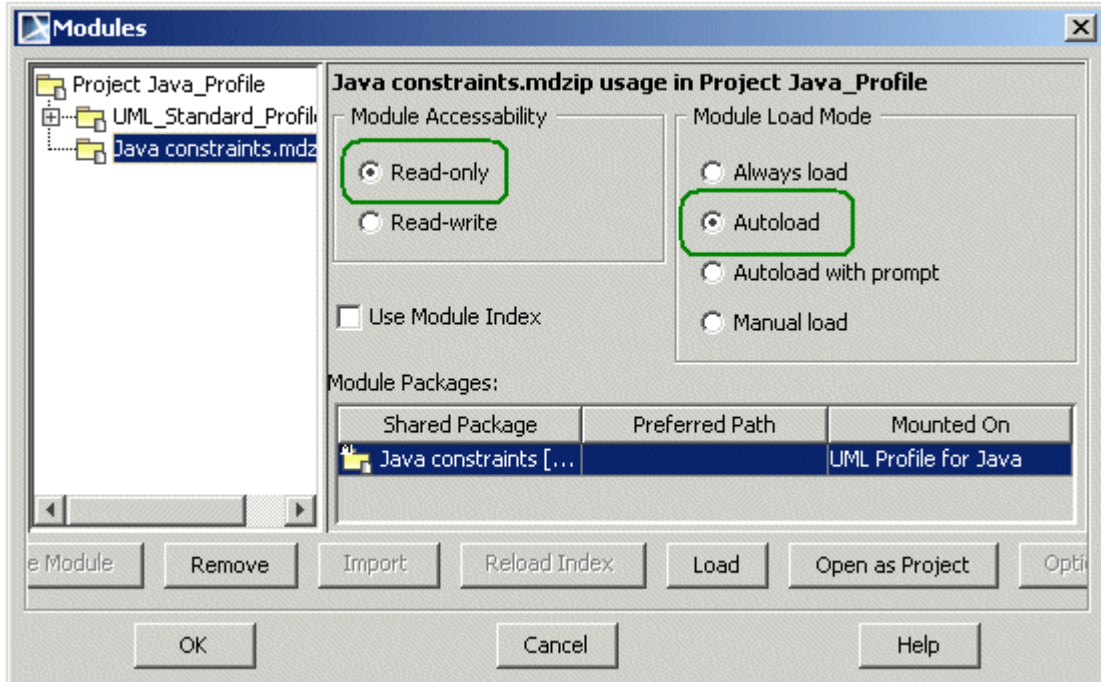


Figure 306 -- Java constraint validation suite module is mounted onto the project as Read-only and Autoload. In this way constraints are not added into project by default, until the validation suite is used.

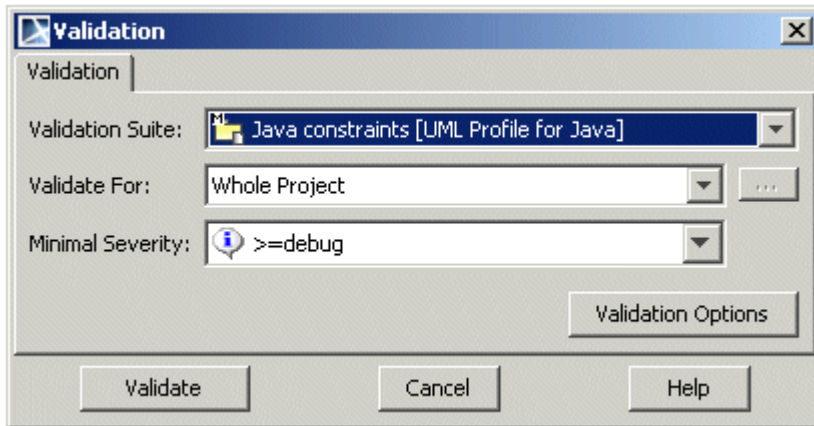


Figure 307 -- Unloaded module with Java constraints validation suite is available through Validation dialog.

Advanced Topics

Global validation rules

Some of the validation rules, specified in OCL, do not refer to the current element (*self*). Such rules are often encountered when using `allInstances()` method to refer to all instances of a particular classifier. Such validation rules are called global validation rules in MagicDraw terminology. Evaluating these rules for each model element is pointless, hence they are evaluated only once per entire validation run. When reporting violations, such rules have a string `<model>` in the column of violating model elements. This means that it is not the concrete element that violates the rule, but the entire model itself.

Here is an example of such a rule (always fails):

```
context anything inv:
false
```

Such a rule is not very useful indeed. The following is another example:

```
context SomeSingletonClass inv:
SomeSingletonClass::allInstances()->size() <= 1
```

This rule checks that there is at most one instance of the *SomeSingletonClass* in the model. The following is a more complex example:

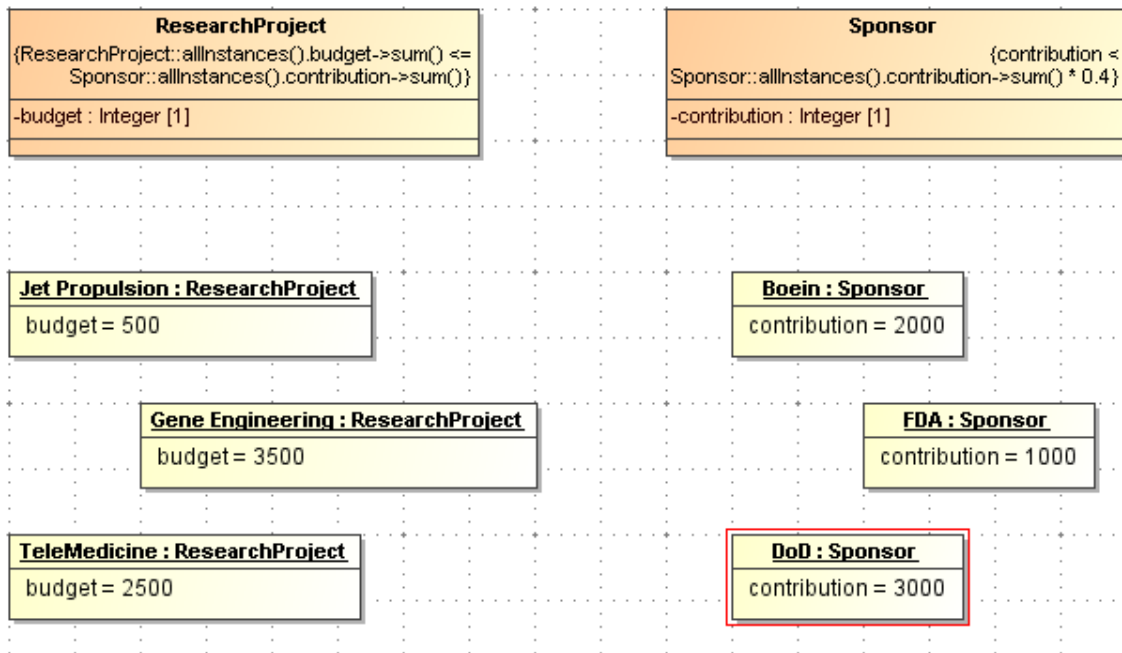


Figure 308 -- example of global validation rule

Here, *ResearchProject* class has a following validation rule (budget must be balanced – sum of expenses of all projects must be less than sum of sponsor contributions):

```
context ResearchProject inv balanced_budget:
ResearchProject::allInstances().budget->sum() <=
Sponsor::allInstances().contribution->sum()
```

Sponsor class has a following rule (anticorruption rule - each sponsor can not contribute more than 40% of the funds):

```
context Sponsor inv anticorruption_law:
contribution < Sponsor::allInstances().contribution->sum() * 0.4
```

Now look at the results of applying these rules:

Element	Severity	Abbreviation	Error Message
<model>	error	BDG	Budget not balanced - overbudget by 500\$
DoD:Classifiers Demo:Sponsor [Classifiers Demo]	error	Sponsor	Each sponsor contribution must be <40% of all contributions

Figure 309 -- results of running the global validation rule

We see that budget balancing rule is a global rule – it is not the concrete instance of *ResearchProject* that violates the rule, but the entirety of instances in the model. Hence the string `<model>` in the column of that offending elements.

However, note that anticorruption rule is not a global rule - it refers to *contribution* field, which is really a shorthand for *self.contribution*, hence this rule refers to self variable and therefore, is not global and is evaluated for each instance of the *Sponsor* class separately.

MagicDraw has no means to determine if the binary validation rules are global, hence all binary rules are treated as local.

Expressions in error messages

When specifying error messages, more than a simple error string can be entered. Error messages can have template areas, which hold expressions, that will be evaluated and expanded when displaying validation results. Refer to the *ResearchProject/Sponsor* example above. The validation result shows:

Budget not balanced – overbudget by 500\$

Where does the number 500 come from? It is not directly specified in the error message string (since it is different for different models) but a calculated value of the expression, embedded in an error string. This error string in this case is:

```
Budget not balanced - overbudget by {  
ResearchProject::allInstances().budget->sum() -  
Sponsor::allInstances().contribution->sum()}$
```

Expressions are embedded in the error messages by using curly brackets - {}. Everything between them is treated as an expression and evaluated for each validation result. Expressions are treated as OCL2.0 expression by default, however you can also use binary expressions. In this case use {bin: <binary expression>} syntax.

Modeling other types OCL2.0 constraints/expressions

Only **inv** constraints can be evaluated in MagicDraw. However, there are more constraint types defined in the OCL2.0 specification. There are also **def**, **init**, **derive**, **pre**, **post**, **body** constraints. These constraints are not evaluated, but can be modeled for documentation purposes. Here is how to model them:

- **def** – create a usual constraint, but apply <<definition>> stereotype, instead of <<invariant>> or <<validationRule>>.
- **init** – place an opaque expression in the defaultValue field of the property.

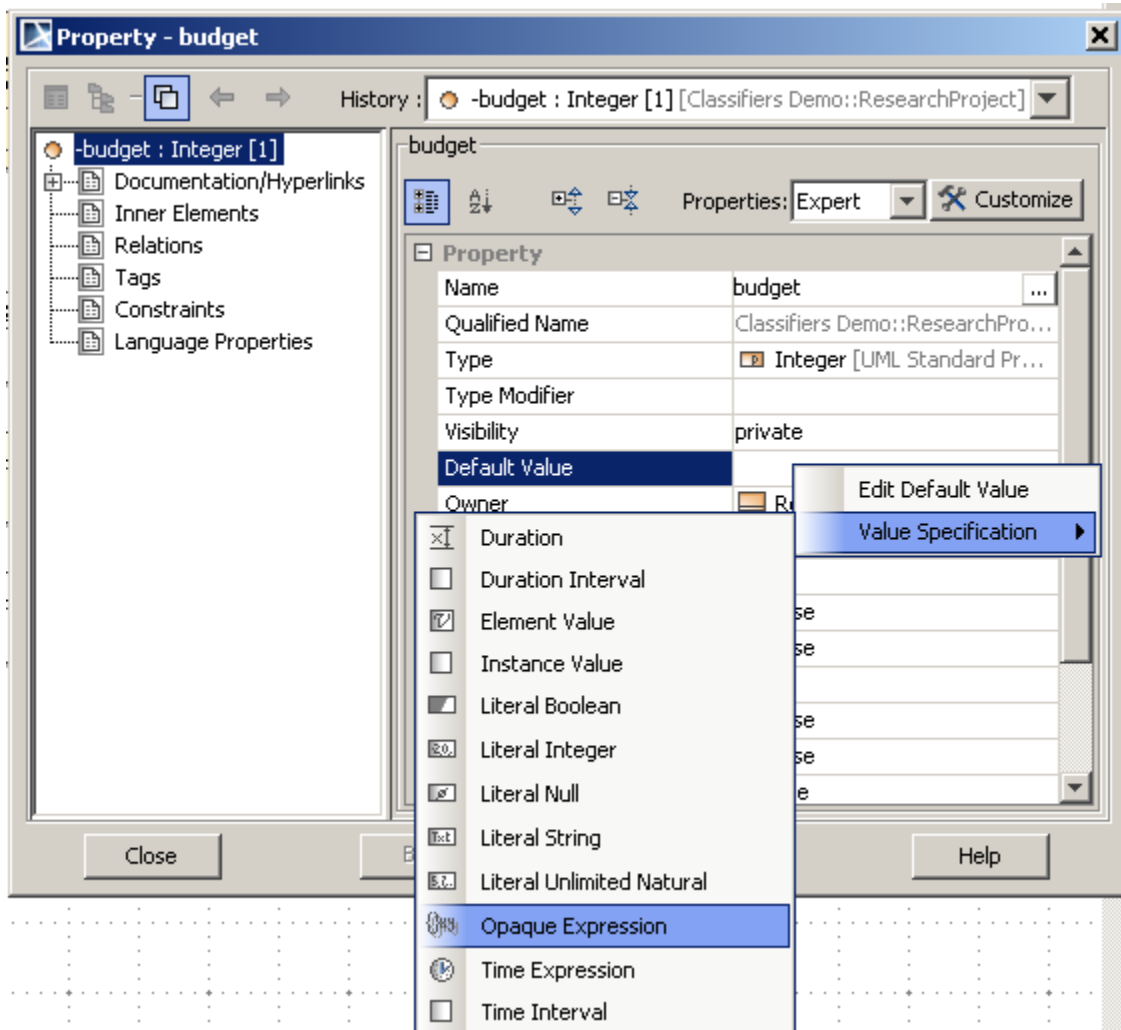


Figure 310 -- setting the expression as a default value of property

To place an opaque expression in the default value of the property, rightclick on the default value field in the specification of the property and select *Value Specification, Opaque Expression*.

- **derive** when the expression is placed in the default value of the property (the same as for **init** expressions) but the property is marked as derived in the specification.

- **pre, post, body** when constraints are placed in the appropriate fields of operation (precondition, postcondition and body condition respectively).

Note that **derive** expressions can be evaluated indirectly, when the validation rule (**inv** constraint) is referencing the property and the validation rule is evaluated.

Unsupported OCL2.0 features

Not all OCL2.0 features are supported in the current release of MagicDraw.

In particular these features are not supported:

- Distinction between null values and undefined values.
- Tuples.
- All the operations, defined in the UML2 superstructure specification on the metaclasses are not present and are not callable.
- Defining and calling operations on classifiers.

There may be some other features that are not working properly. These features can be reported to support@magicdraw.com. MagicDraw uses external library – Dresden OCL Toolkit for constraint evaluation.

Adding/customizing severity levels

If the default severity level choice, provided by MagicDraw is not enough for you, new severity levels can be added. This can be done by editing the *SeverityKind* enumeration in the *UML Standard Profile::Validation Profile* package. Each enumeration literal in this enumeration corresponds to available severity levels. Severity levels in this enumeration should be sorted in ascending order.

If you need to specify a new icon for your custom severity level:

1. Create stereotype, derived from the *imaged* stereotype, with the *EnumerationLiteral* as base class.
2. Set the necessary icon on this stereotype.
3. Apply this stereotype on your custom severity level enumeration literal.
4. Additionally, specify the *highlightColor* tagged value on the literal. This field (of the *String* type) determines how the offending elements will be highlighted in the diagrams. The string format is the same as for the specifying colors in HTML pages (as described in <http://www.w3.org/TR/html4/types.html#h-6.5>). Simple string con-

straints (such as *highlightColor="red"*) or numeric values (such as *highlightColor="#FF0000"*) can be used here.

Performance Issues

When validation rules, written in OCL are evaluated, MagicDraw generates Java source for them and invokes Java compiler to compile them into an executable form. Hence, on the first run of validation there is usually a delay of 20-30 seconds (depending on the computer performance) while Java compiler is loading. Subsequent runs will be faster than the first one.

Also, this process consumes some amount of RAM. If the validation process is run heavily on medium-large projects, increasing the default Java VM size is advisable. By default, VM size is set to 400MB in MagicDraw; increasing this to 600 (or 800 if the computer has sufficient RAM) might improve the performance.

Active Validation

[View Online Demo](#) Active Validation

NOTE This functionality is available in all MagicDraw editions.

Active Validation instantly checks the accuracy, completeness, and correctness of a model, displays errors in the model, and suggests solutions.

The following modeling cases are validated with the Active Validation:

- Parameters and arguments synchronization validation
- Correct ownership validation
- Orphaned proxies' validation.

Active Validation is an extendable mechanism that identifies common problems and solutions. Custom validation suites and constraints can be created using binary or OCL constraint. Invalid symbols are marked on diagrams and elements in the model.

Errors in the model are represented in the following ways:

- If an invalid or incomplete model is created, an error indicator will appear in the bottom right corner of MagicDraw.
- Invalid elements are marked in the Browser and Diagram.

From the invalid element/symbol shortcut menu, you can analyze incorrect elements, and solve problems, resulting from the errors, through the **Active Validation Results** window.


Model is validated with three predefined validation suites and the Validation is performed for the parameters and arguments synchronization, correct ownership, and orphaned proxies. You can modify these suites and create your own through the **Validation Options** dialog box.

Detecting errors in the model

When Active Validation detects errors in the model, errors will be revealed in the following ways:

- the failure indicator
- in the Browser, an invalid element is marked with little cross and x symbol.
- on the diagram, invalid symbol is highlighted.

Failure indicator

If an error occurs, the failure indicator  will appear in the bottom right corner of MagicDraw (see Figure 311 on page 631). Click this indicator and the **Active Validation Results** window will open.

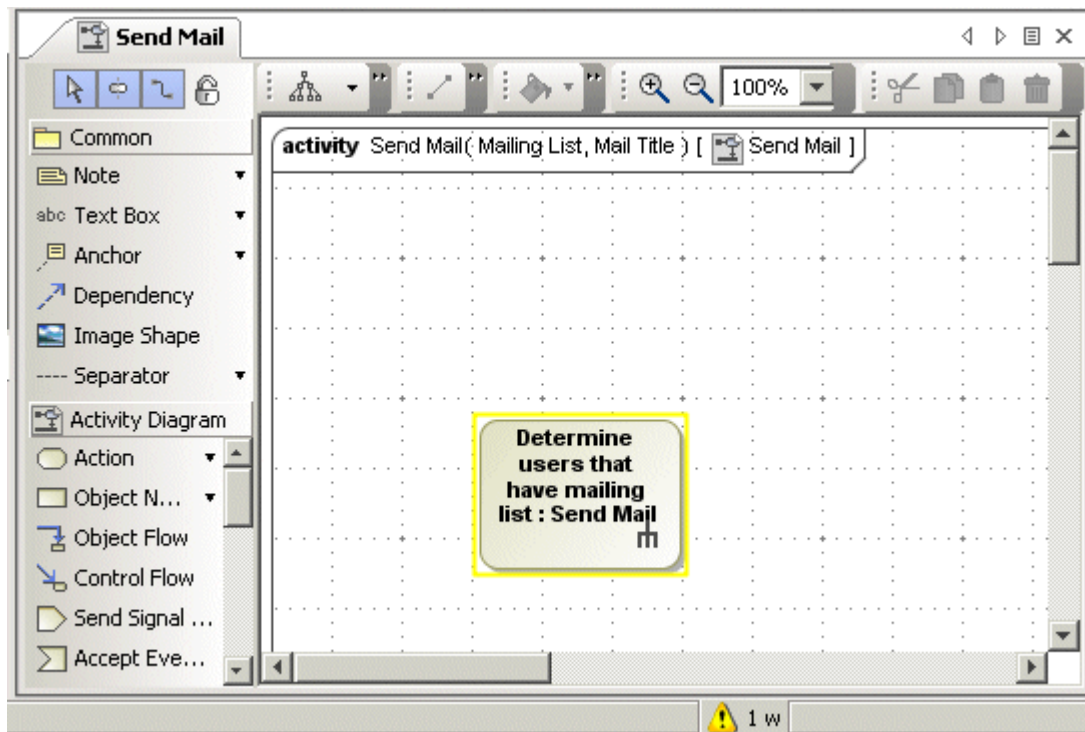


Figure 311 -- General validation notification in MagicDraw

See the parts of the failure indicator in the figure below.

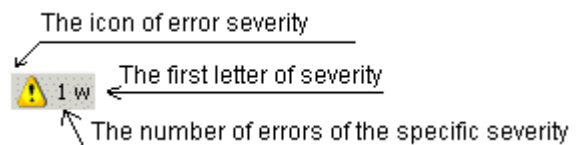


Figure 312 -- The failure indicator





Example of the failure indication	Explanation	Possible values
	Error symbol showing the level of severity.	 - indicating warning  - indicating error or fatal error.  - indicating debug error or info.
1	Number of errors of that specific severity	1, 2, 3 ...
w	First letter of the error severity	F - fatal error E - error W - warning D - debug I - info

TABLE 4. Parts of the failure indicator

Marking errors in the Browser

An invalid model element is marked in the Browser with a small x symbol (see Figure 313 on page 633). The owner of this element is marked with a small grey symbol.

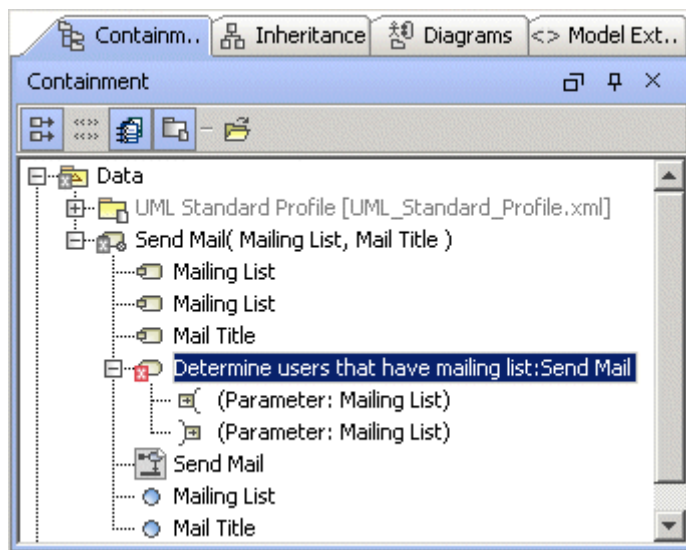


Figure 313 -- Invalid elements marking in browser

Highlighting errors on the diagram

The element symbol is colored according to the severity of the error on the diagram.

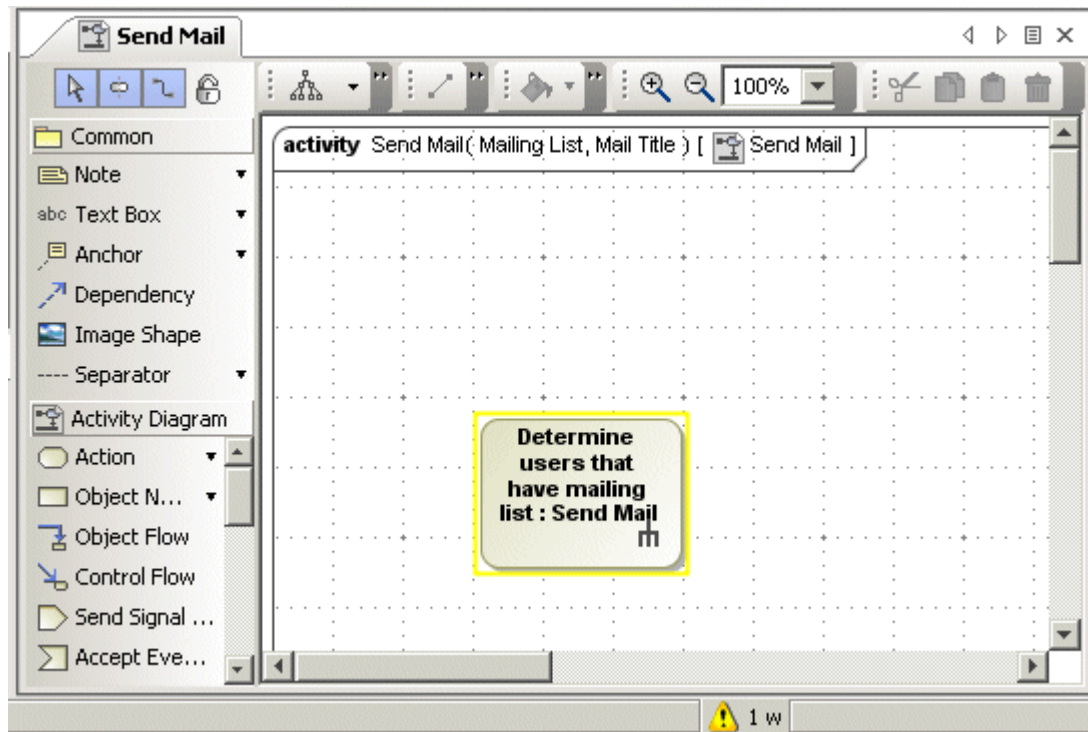


Figure 314 -- Invalid symbols marking in diagram

Handling incorrect model

The **Active Validation Results** window lists the active validation results. It can navigate users to have ability not only to navigate to constraints and invalid elements or symbols, correct errors, filter, and ignore problems.

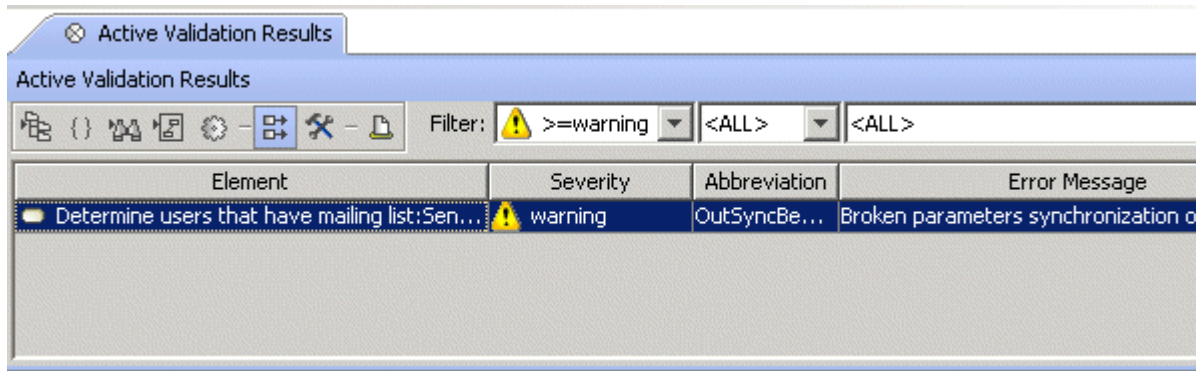


Figure 315 -- The Active Validation Results window

Changing the Active Validation Options

To change the active validation options:

1. From the **Options** main menu, select **Project**.
2. In the **Project Options** dialog box, select the **General project options** group.
3. In the **Active Validation** group, specify the active validation options.

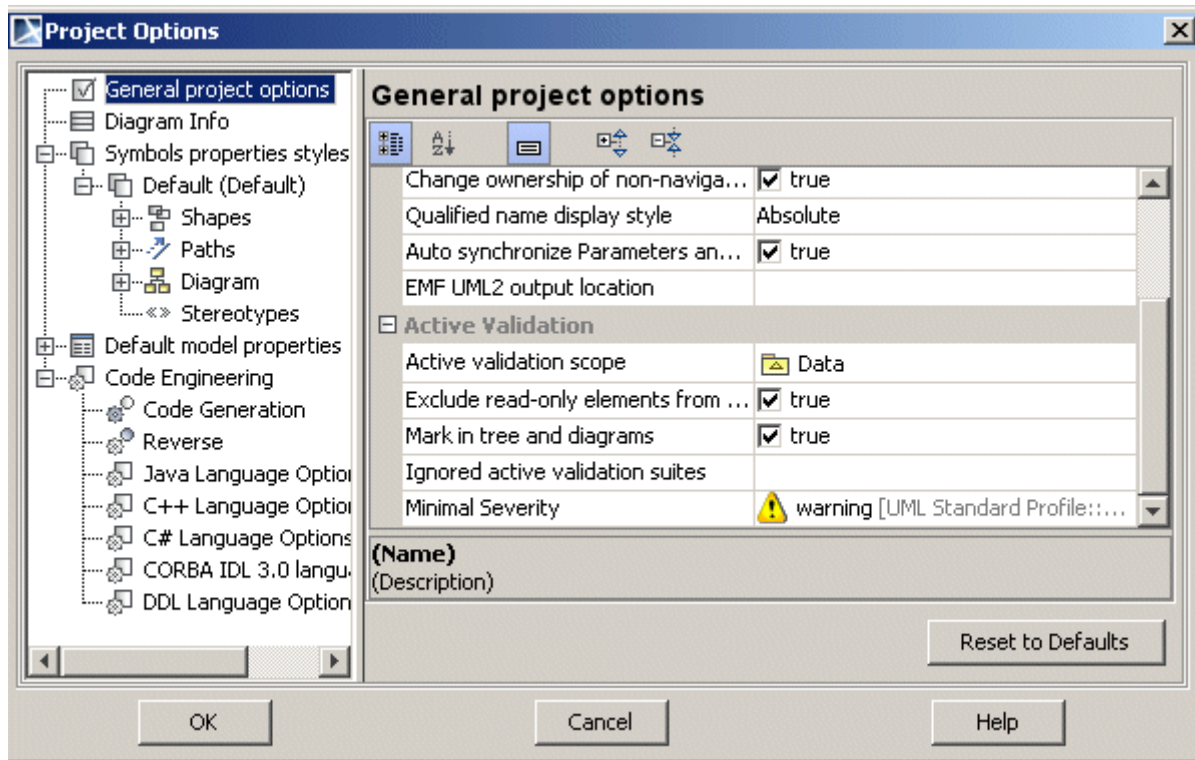


Figure 316 -- The Project Options dialog box, Active Validation group

The Active Validation Suites

Validating Parameters and Arguments Synchronization

More information about Parameter Synchronization can be found here “Parameters synchronization with Arguments” on page 960.

In most cases the parameters and arguments synchronization is not visible. Automation and synchronization between Parameters and Arguments (for example Operation parameters and Call Behavior Action Pins) increase modeling speed and helps avoid modelling errors.

Arguments should always in sync with Parameters. But in cases when synchronization is not possible or corrupted, the active validation will notify the user by highlighting the symbols on the diagrams.

Shape Ownership

The term “symbol” means a visual representation of some model elements in the diagram. Symbols are further subdivided into shapes and paths (lines in the model, for displaying various relationships).

When drawing UML diagrams, the element ownership is not easily visible. One diagram can contain symbols for elements from several different packages. Element rearrangements in the model may lead to situations where the element ownership in the model does not match the symbol ownership as displayed in the diagram. Such situations are not easy to detect from diagram view.

MagicDraw version 15.0 has a built-in validation code to detect this mismatch. This feature is enabled by default and run unnoticed without requiring any additional input from the user.

When the symbol ownership on the diagram pane does not match the actual element ownership in the model, the symbol is highlighted with red. So, you will easily see it on the diagram and will be able to correct the problem (problem correction hints are also suggested).



Figure 317 -- Class symbol highlighted with red border

When the mismatch is resolved, the highlighted symbol will return to normal.

The symbol ownership validation feature uses the same mechanism for problem highlighting as the generic validation feature available in MagicDraw Enterprise edition. If you run some validation suites against the model, the element can be highlighted due to any of the validations failures:

- either validation rule(s) from that suite,
- or this automatic symbol ownership checking rule(s).

For more information about validation, see “Validation” on page 601.

Symbol ownership validation covers two cases:

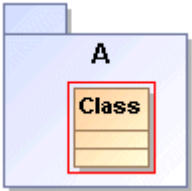

- Class diagram and its derivatives (use case, implementation, composite structure diagrams etc.) can display elements from many model locations and show their

ownership. If shape owner is incorrect (not the same as the element owner in the model), this shape will be highlighted.

- Dynamic diagrams (state machine, activity and interaction diagrams) have a restriction that only elements from one definite state machine, activity or interaction can occur in each concrete diagram. Validation rule checks for these diagrams that only elements from correct state machine, activity, or interaction appear in the diagram. All foreign elements are highlighted as erroneous.

These cases are summarized in the table below

	Validation of ownership on the diagram pane	Validation of diagram owner, on which element is drawn
Validation rule	Check if the symbol owner on the diagram correctly reflects the element owner in the model.	Check if the element and the diagram on which the symbol is drawn belong to the same owner. This is valid for dynamic diagrams. The checking is performed according to the context of the diagram.
What is checked	Shapes are checked, excluding paths.	Validation works for shapes, including paths.
Cases	<p>Check is performed:</p> <ul style="list-style-type: none"> • if the element symbol is draw on the same package/model/profile to which it actually belongs; • if the element symbol is drawn on the same component/node to which it actually belongs; • if the element symbol is drawn on the same state to which it actually belongs; <p>and other similar cases.</p>	<p>Check is performed:</p> <ul style="list-style-type: none"> • if the communication diagram and elements from the communication diagram are in the same interaction; • if the sequence diagram and its elements are in the same interaction; • if the state diagram and its elements are in the same state machine; • if the protocol state machine diagram and its elements are in the same protocol state machine. • if the activity diagram and its elements are in the same activity.

	Validation of ownership on the diagram pane	Validation of diagram owner, on which element is drawn
Examples	For example, on the diagram pane, an element shape is nested in package A, but actually the element is in package B.	For example, a diagram belongs to activity A, but elements of this diagram belong to activity B.
		

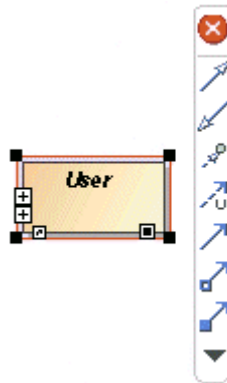
Important Ownership validation is not available in the Business Process diagram.

- TIP** To locate the actual owner of the element:
- Select the element on the diagram pane and right click. Select the **Select in Containment Tree** command. In the Browser you can see the actual element. The parent in the tree is the actual owner.
 - In the element **Specification** dialog box, look up the **Owner** property.

Solving the detected symbol ownership problems

In this section you will find out the reasons why an element is highlighted in red and how to solve symbol ownership problems quickly.

Select the highlighted element on the diagram pane. The Smart Manipulator toolbar opens. Notice the red button on the top of Smart Manipulator toolbar.

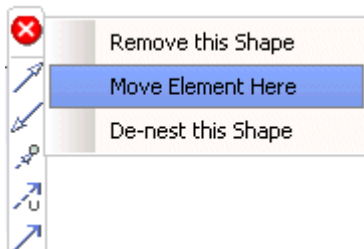


To see the reason why a shape is highlighted, move the mouse pointer over the red button in the Smart Manipulator toolbar. You will see a tool tip that explains why the element is highlighted. For example: “Shape ownership in the diagram does not correspond to the element ownership in the model”.

You may either solve the problem yourself or choose one of the suggested solutions. To select an available solution, click the red button in the Smart Manipulator toolbar. A menu with the following commands will open (note that some items might not be included depending on the situation in the model):

- **Remove this Shape.** Symbol is deleted from the diagram pane.
- NOTE:** When a symbol is deleted, the element is not deleted from the project.
- **Move Element Here.** Element is moved to the new owner in the model, so that the element ownership in the model corresponds to the present shape ownership in the diagram.
 - **De-nest this Shape.** Shape (not element!) will be extracted from the current owner and placed directly on the diagram. This solution is applicable only for some cases

(e.g. De-nest, this Shape command is not available for shape, placed in the incorrect state machine, activity, interaction diagrams).



To turn the ownership validation on/off

- Select/clear the **Validate Shape Ownership** check box in the **Analyze** main menu.
- Select/clear the **Validate Shape Ownership** check box in the **Environment Options** dialog box, **Diagram** branch, **Display** options group.

To define the ownership checking period

In the **Environment Options** dialog box, **Diagram** branch, **Display** group, **Diagram Checking Period** text box, type the diagram checking period in seconds.

Related Topics

“Setting Environment Options” on page 129.

“Smart Manipulation” on page 269.

UML model correctness

Active validation instantly check the most important correctness rules of UML model. The following validation rules are checked: Ports compatibility, Pin types compatibility, Slot and Tags multiplicity correctness and others. Automated solutions are suggested for solving the model errors.

Validating the Orphaned Proxies (OP)

Checking for OP moved from a separate feature into one unified validation mechanism that identifies problems and solutions.

During model editing theres is possibility can happen that an element is referenced using proxy, but the real element is not shared or does not exist in the module. After the modules that load such proxies are detectd, they are called Orphaned Proxies. These proxies are marked by a validation sign in the browser. Actions to clean such proxies are added.

Customizing the Active Validation

Any constraint (binary or OCL), additionally optimized, can be used to validate models in real-time. MagicDraw provides predefined suites for validation of: Correct ownership in the model and on the diagram, Parameters and arguments synchronization, missing referenced elements (Orphaned Proxies) in the modules, and others.

A model is validated automatically without any additional input. Additional constraints can be added or properties can be customized from **Analyze** (main menu)-> **Validation**.

Validate element that has no representation in diagram

Element or its symbol is highlighted in diagram pane and in Browser if it is owner of element that can not be represented in diagram and has validation error.

See the examples of the spelling error in the table below:

Sample description	Representation in Diagram	Representation in Containment Tree	Notes
Use Case documentation (comment owned by Use Case and annotating Use Case) has a spelling error	Use Case symbol is highlighted with red dashed border (see Figure 318 on page 643).	The owner of the hidden element with error is marked with white x in red quadrant in the Containment tree as itself would have error (see Figure on page 643).	To the symbol smart manipulator toolbar there is added additional button (grey circle with cross inside) that presents the errors solving solutions.

Sample description	Representation in Diagram	Representation in Containment Tree	Notes
Use Case extension point name has a spelling error	Use Case symbol is highlighted with red dashed border (see Figure 319 on page 644).	The owner of the corrupted and not represented in diagram element is marked with white X in grey quadrant in the Containment tree (see Figure 320 on page 644).	

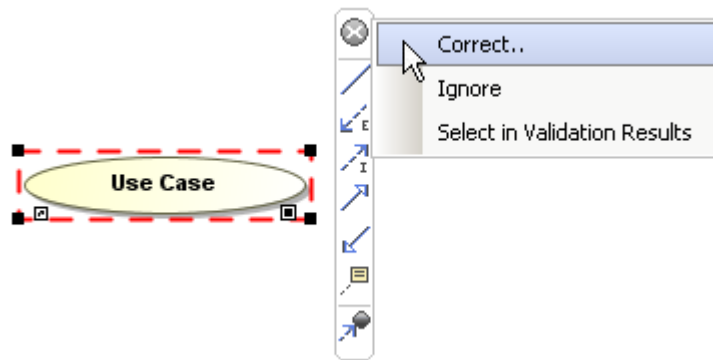
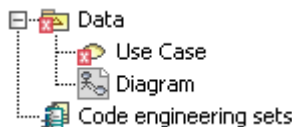


Figure 318 -- The Use Case is highlighted with red dashed border, because it has documentation with spelling error



The Use Case is highlighted with white X in red quadrant in the Containment tree, because it has documentation has a spelling error

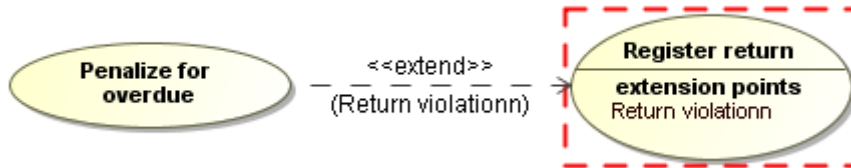


Figure 319 -- The "Register return" Use Case is highlighted with red dashed border, because the "Return violationnn" Extension Point has a spelling error

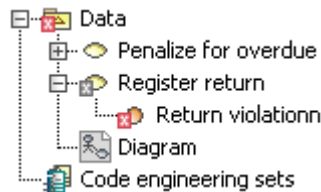


Figure 320 -- The "Register return" Use Case is highlighted with white X in grey quadrant in the Containment tree, because the "Return violationnn" Extension Point has a spelling error

Usage in Diagrams

Definition The term “symbol” means a visual representation of some model element in the diagram. Symbols are further subdivided into shapes and paths (paths are lines used in the model for representing various relationships).

The symbol usage in diagrams functionality allows the user to see the usage of a symbol throughout the diagrams of a project. The two diagram searching possibilities are:

- "Searching for symbol usage in diagrams from the element specification dialog box"
- "Searching for symbol usage in diagrams from the element shortcut menu".

Searching for symbol usage in diagrams from the element specification dialog box

1. To see the diagram list, open the element specification dialog box and select **Usage in Diagrams**. The table shows all diagrams in which the symbol is represented.
2. Select the diagram you want to open.
3. Press the **Open** button. The diagram is opened and the symbols of the element are selected. If the diagram includes more than one of the same element symbol, all symbols of the same element are selected in the diagram.

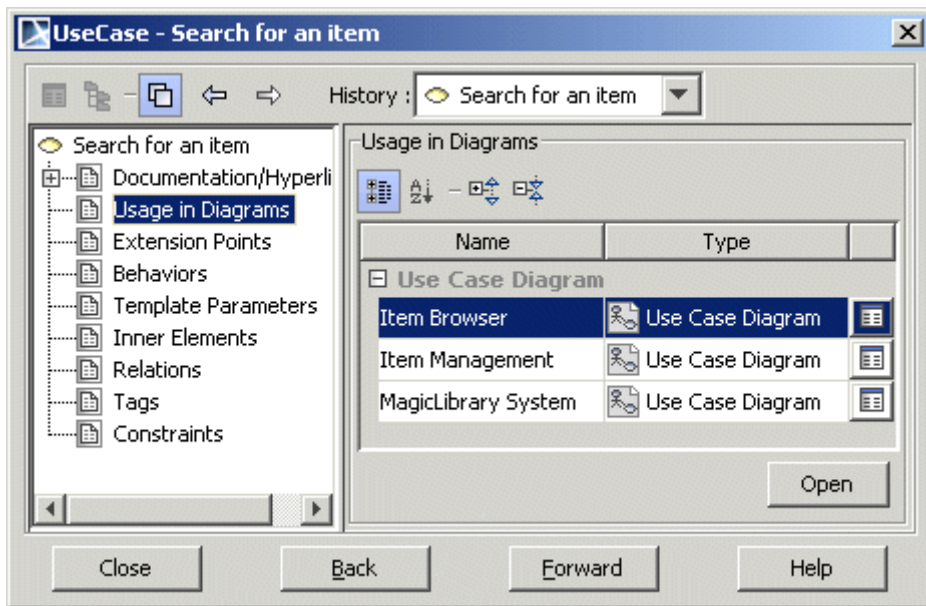


Figure 321 -- The element specification dialog box, the Usage in Diagrams branch

To open the diagram specification dialog box, press the  button near the diagram in the list.

For more information about working with the element specification dialog box, see "Specification dialog boxes" on page 325.

Searching for symbol usage in diagrams from the element shortcut menu

1. Select the element in the Browser or select the symbol in the diagram.
2. From the element shortcut menu, select **Go To** and then **Usage in Diagrams**.
3. Select the diagram that you want to see the particular element symbol. The diagram is opened and symbols of the current element are selected on the diagram.

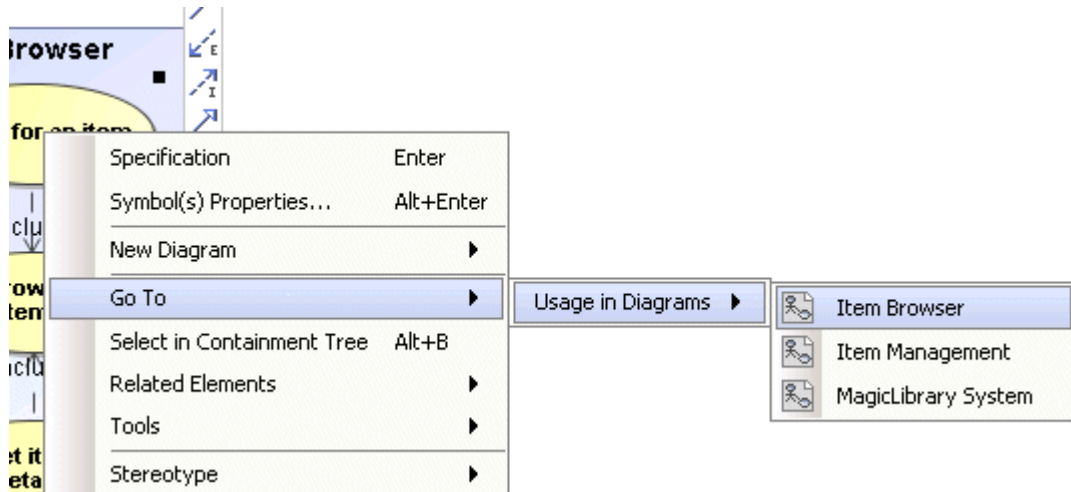


Figure 322 -- Searching for symbol usage in diagrams from the element shortcut menu

Tip!

You can also search from usage in diagrams in the following way:

- Select the symbol on the diagram. Then, from the **Analyze** main menu, select the **Go To** command and the **Usage in Diagrams**.

NOTES

- All symbols of the same element are selected in the opened diagram.
- If more than one symbol exists in the same diagram, then the diagram is zoomed out, to fit the view in screen.

For information about the Usages /Dependencies functionality for analyzing associations between elements, see "Dependencies analysis" on page 557.

9 UML DIAGRAMS

In software development, the diagram is the equivalent of a blueprint. To meet the various needs of many parties, we often need several different “blueprints” of the same system. Furthermore, every system is described by many different aspects. For example:

- Functional (static structure and dynamic interactions)
- Nonfunctional (timing requirements, reliability, and deployment)
- Organizational (work organization and mapping to code modules)

MagicDraw supports the following diagrams that are defined in UML 2:

UML Diagrams	MagicDraw UML Diagrams
Use Case Diagram	Use Case Diagram ^a
Class Diagram	Class Diagram ^a
Object Diagram	
Composite Structure Diagram	Composite Structure Diagram
State Machine Diagram	State Machine Diagram
Protocol State Machine Diagram	Protocol State Machine Diagram
Activity Diagram	Activity Diagram
Interaction Overview Diagram	Interaction Overview Diagram
Sequence Diagram	Sequence Diagram
Communication Diagram	Communication Diagram
Component Diagram	Implementation Diagram ^a
Deployment Diagram	

a. The Package Diagram is provided for use with Use Case, Class, and Implementation Diagrams

Architectural Views

UML defines 13 diagrams that describe 4+1 architectural views:

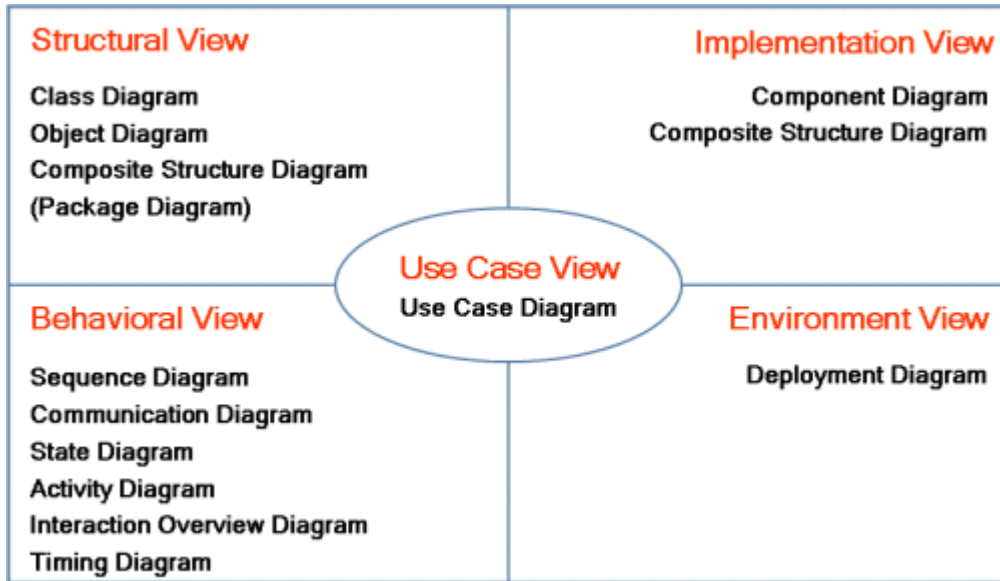


Figure 323 -- Architectural Views

Several kinds of diagrams provide a visual notation for the concepts in each view.

Use Case View

The use case view represents the functionality and behavior of a system or subsystem as it is perceived by external users. This view is targeted mainly at customers, designers, developers, and testers.

The use case view usually is presented as a number of use cases and actors in Use Case diagrams. Occasionally it is used in Activity and Sequence diagrams.

The use case view is central because the contents drive the development of the other views. It is also used for project planning. Every single use case unit is deemed as a manageable unit during the project execution.

Structural View

The structural view represents structural elements for implementing a solution for defined requirements. It identifies all of the business entities and how these entities are related to each other. Usually entities are represented as classifiers and their instances in class and object diagrams in multiple abstraction levels. System decomposition to different layers can be displayed using Package diagrams. A Composite structure diagram can be used to represent the classifier inner structure. The system structural view artifacts are created by software architects and represent the system implementation design solutions.

Behavioral View

The dynamic behavior of the system is displayed on the Interaction (sequence and collaboration), State, Activity, Interaction overview, and Timing diagrams. It focuses mainly on the interactions that occur between objects inside a system, activities and work performed by the various parts of a system, and state changes within a particular object or collaboration. Rather than defining the participants of the system, it defines how particular use cases are executed, which provides value for the external user. The dynamic view is concerned about what is happening inside the system and how those actions impact other participants.

Implementation view

The implementation view describes the implementation artifacts of logical subsystems defined in the structural view. It may include the intermediate artifacts used in a system construction (code files, libraries, data files, etc.) This view defines dependencies between the implementation components and their connections by the required and provided interfaces. Components and their relationships are displayed on the Component diagram. Inner parts of the component can be represented with the Composite structure diagrams. The implementation view helps analyze system parts and their dependencies in a higher component level.

Environment view

The environment view represents the physical arrangement of a system, such as computers and devices (nodes) and how they are connected to each other. In contrast to the component view, the deployment view is concerned with the physical structure of the system and the location of the software modules (components) manifested by artifacts within the system.

The environment view is displayed on the deployment diagram that is called the implementation diagram in MagicDraw™ UML.

Class Diagram

A class diagram is a graphic representation of the static structural model. It shows classes and interfaces, along with their internal structure and relationships. The classes represent types of objects that are handled in a system. A class diagram does not show temporal information, it describes only the classification. The instances of those types (objects) are instantiated only on the runtime and are represented by an object and the interaction diagrams.


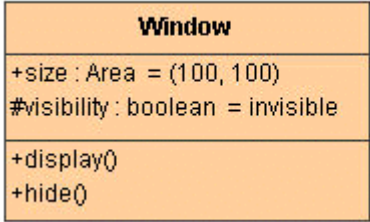

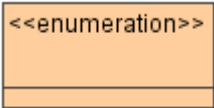




The classes can be related to each other in a number of ways: associated (connected to each other), dependent (one class depends/uses another class), specialized (one class is a subtype of another class), or packaged (grouped together as a unit – package). A class diagram does not express anything specific about the relationships of a given object, but it does abstractly describe the potential relationships of one object with other objects.

A system typically has a number of class diagrams – not all classes are inserted into a single class diagram. A class may have multiple levels of meaning and participate in several class diagrams.

A class diagram is the logical map of an existing or future source code.







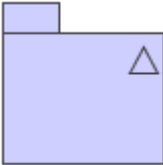
The classes can be grouped into packages. The packages can be nested within other packages. A package, as an entity, may have all the relationships that can be drawn for a class. Those relationships are derived from the classes or packages that are nested within two particular packages (i.e., the relationship between packages reflects a set of relationships between classes placed in those packages).


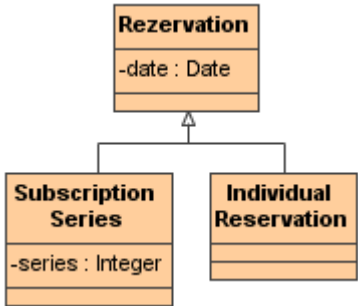

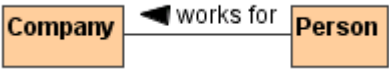


Class diagram elements

Model element	Button (hot key)	Notation
Class A descriptor for a set of objects with similar structures, behaviors, and relationships.	 (C)	 <div> <div>class name</div> <div>attributes</div> <div>operations</div> </div>
Enumeration A user-defined data type whose instances are a set of user-specified named enumeration literals. The literals have a relative order but no algebra is defined on them.	 (K)	
Class by Pattern	 (SHIFT+P)	
Signal		
Data Type		
Primitive Type		

9 UML DIAGRAMS


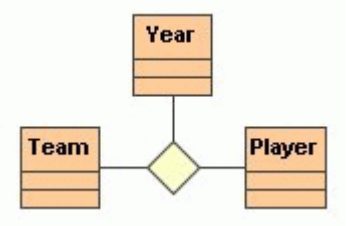

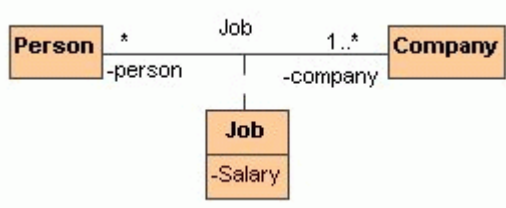

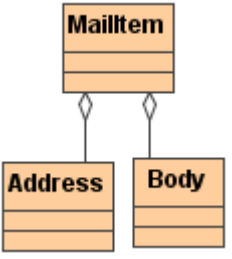

Class Diagram


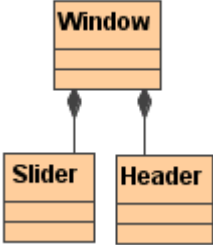


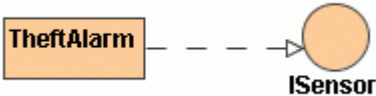

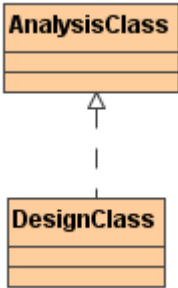
Model element	Button (hot key)	Notation
Interface The description of a visible behavior of a class, a component or a package. Attributes and operations inside the Interface can be suppressed.	 (I)	  UpdatePrices
Package A group of classes and other model elements.	 (P)	
Model A model is an abstraction of a physical system from a particular point of view. A model contains a hierarchy of packages/sub-systems and other model elements that describe the system.	 (M)	





Model element	Button (hot key)	Notation
Generalization A relationship between a more general and a more specific element. Note: Choose a different Generalization direction from the toolbar to draw a line with an opposite arrow end.	 (G)	 <pre> classDiagram class Rezervation { -date : Date } class SubscriptionSeries { -series : Integer } class IndividualReservation { } Rezervation < -- SubscriptionSeries Rezervation < -- IndividualReservation </pre>
Association A connection among classes, which also means a connection among objects of those classes.	 (S)	 <pre> classDiagram class Company class Person Company --> Person : works for </pre>
Directed Association		
Non-navigable Association		







9 UML DIAGRAMS






Class Diagram

Model element	Button (hot key)	Notation
N-ary association An association among two or more classes (a single class may appear more than once).	 (O)	
Association Class The Association Class is a declaration of a semantic relationship between Classifiers. The Association Class, which has a set of features of its own, is both an Association and a Class.		
Aggregation An aggregation is an association that represents a whole-part relationship.	 (A)	
Directed Aggregation		

Model element	Button (hot key)	Notation
Composition A composition is a form of aggregation with a stronger ownership and coincident lifetime of part with the whole.	 (F)	
Directed Composition		
Interface Realization A relationship is usually used between an interface and an implementation class. Note: Choose a different Interface Realization direction from the toolbar to draw a line with an opposite arrow end.	 (R)	
Realization A relationship between a specification and its implementation.	 (E)	

Model element	Button (hot key)	Notation
Substitution A substitution is a relationship between two classifiers.		
Usage A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation. Note: Choose a different Usage direction from the toolbar to draw a line with an opposite arrow end.		
Abstraction An abstraction is a dependency relationship that relates two elements or sets of elements that represent the same concept at different levels of abstraction or from different viewpoints.		
Template Binding A binding is a relationship between a template and a model element generated from the template.	 (B)	

Model element	Button (hot key)	Notation
Package Merge A package merge is a directed relationship between two packages that indicates that the contents of the two packages are to be combined.		
Package Import A package import is defined as a directed relationship that identifies a package whose members are to be imported by a namespace.		
Element Import An element import is defined as a directed relationship between an importing namespace and a packageable element.		
Instance	 (SHIFT+O)	
Link A connection between two or more objects.	 (SHIFT+L)	 <pre> classDiagram class MainWindow[":MainWindow"] class ClientWindow[":ClientWindow"] MainWindow -- ClientWindow </pre>
Profiling Mechanism Toolbar		

Model element	Button (hot key)	Notation
Stereotype A stereotype is an extension mechanism that defines a new and more specialized element of the model based on an existing element.	 (SHIFT+S)	
MetaClass A class whose instances are classes. Meta-classes are typically used to construct metamodels.		
Extension An extension is used to indicate that the properties of a metaclass are extended through a stereotype, and gives the ability to flexibly add (and later remove) stereotypes to classes.		
Profile A Profile is a kind of Package that extends a reference metamodel.		
Profile Application A profile application is used to show which profiles have been applied to a package.		

Use Case Diagram

A use case is a description of the functionality (a specific usage of a system) that a system provides. The use case descriptions may exist in a textual form (a simple table), where the use case diagram provides additional information about the relationship between the use cases and the external users. The diagram also allows a definition of the system's boundary.

The Use cases are described only in terms of how they appear when viewed externally by the user (a system's behavior as the user perceives it), and do not describe how the functionality is provided inside the system. The Use cases are not object-oriented, but they are included in the UML to simplify the approach of the project's lifecycle -- from the specification to the implementation.

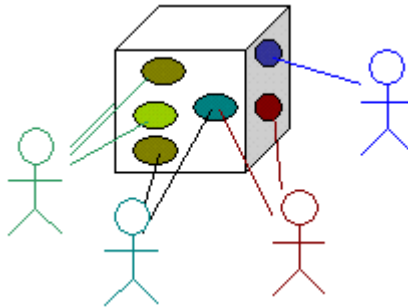







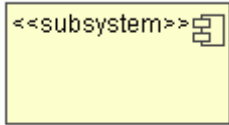



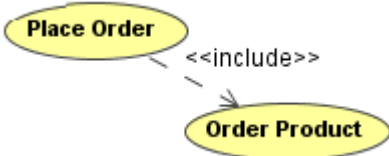





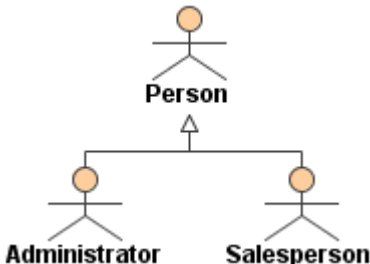


Figure 324 -- The schematic view of the use cases in the system.

Use Case diagram elements

Element	Button (hot key)	Notation
Actor Actors represent roles played by human users, external hardware, and other subjects. An actor does not necessarily represent a specific physical entity but merely a particular facet (that is, "role") of some entities that is relevant to the specification of its associated use cases.	 (A)	
Use Case A use case is a kind of behavior-related classifier that represents a declaration of an offered behavior. Each use case specifies a particular behavior, possibly including the variants that the subject can perform in collaboration with one or more actors. The subject of a use case could be a physical system or any other element that may initiate a behavior, such as a component, a subsystem, or a class.	 (U)	
Package A group of classes and other model elements. A package may contain other packages.	 (P)	

Element	Button (hot key)	Notation
Subsystem A subsystem is treated as an abstract single unit. It groups model elements by representing the behavioral unit in a physical system.	 (Y)	
System Boundary Another representation of a package. A system boundary element consists of use cases related by Exclude or Include (uses) relationships, which are visually located inside the system boundary rectangle.	 (B)	
Include An include (uses) relationship from use case A to use case B indicates that an instance of the use case A will also contain the behavior as specified by B.	 (C)	
Extend A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case. The extension takes place at one or more specific extension points defined in the extended use case. Note: Choose a different Extend direction from the toolbar to draw a line with an opposite arrow end.	 (E)	


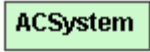




Element	Button (hot key)	Notation
Association The participation of an actor in a use case, i.e. instances of the actor and instances of the use case communicate with each other. This is the only relationship between actors and use cases.	 (S)	
Generalization A relationship between a more general and a more specific element. Note: Choose a different Generalization direction from the toolbar to draw a line with an opposite arrow end.	 (G)	



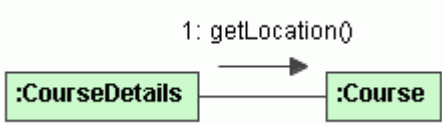




Communication Diagram







The Communication diagram illustrates the various static connections between objects, and models their interactions. It also presents a collaboration that contains a set of instances as well as their required relationships given in a particular context, and includes an interaction that defines a set of messages. These messages specify the interaction between the classifier roles within a collaboration that will serve to achieve the desired result.

A Communication diagram is given in two different forms: at the instance level or at the specification level.

Communication Diagram elements

Element	Button (hot key)	Notation
Lifeline A lifeline represents an individual participant in the Interaction. The Lifelines represent only one interacting entity.	 (O)	
Connector Specifies a link that enables communication between two or more lifelines. Each connector may be attached to two or more connectable elements, each representing a set of lifelines.	 (C)	
Connector to Self Self connector for self-calls. It begins and ends on the same lifeline.	 (S)	






Element	Button (hot key)	Notation
Message to Right Message to Left <p>A Message defines a particular communication between the Lifelines of an Interaction. It implies that one object uses the services of another object, or sends a message to that object. A communication can be formed by e.g. raising a signal, invoking an Operation, creating or destroying an Instance.</p> <p>Click on the arrow to expand the toolbar and choose an appropriate message type.</p>	 	
Call Message to Right Call Message to Left	 	
Send Message to Right Send Message to Left	 	

Element	Button (hot key)	Notation
Reply Message to Right		
Reply Message to Left		
Create Message to Right		
Create Message to Left		
Delete Message to Right		
Delete Message to Left		

Sequence Diagram








A sequence diagram shows the interaction information with an emphasis on the time sequence. The diagram has two dimensions: the vertical axis that represents time and the horizontal axis that represents the participating objects. The time axis could be an actual reference point (by placing the time labels as text boxes). The horizontal ordering of the objects is not significant to the operation, and you may rearrange them as necessary.




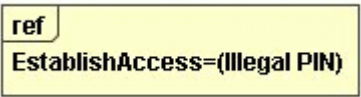

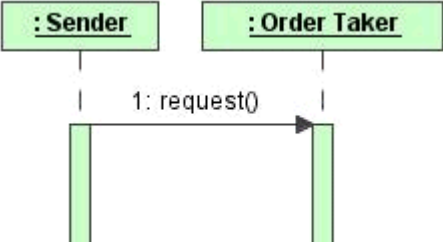


Sequence diagram elements

Model elements	Button (hot key)	Notation
<p>Lifeline Represents the existence of an object at a particular time.</p> <p>Activation Bar Focus of control. Shows the period during which an object is performing an action either directly or through a subordinated procedure.</p>	 <p>(O)</p>	
Alternatives	 <p>(SHIFT+A)</p>	
Loop	 <p>(SHIFT+L)</p>	
Option	 <p>(SHIFT+O)</p>	

9 UML DIAGRAMS




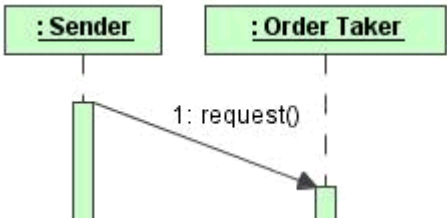
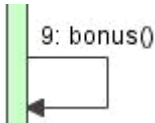
Sequence Diagram


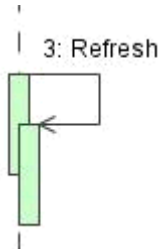

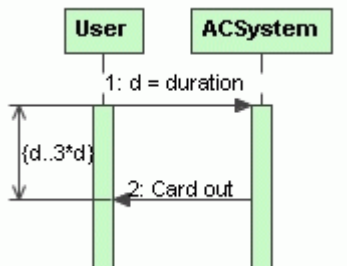

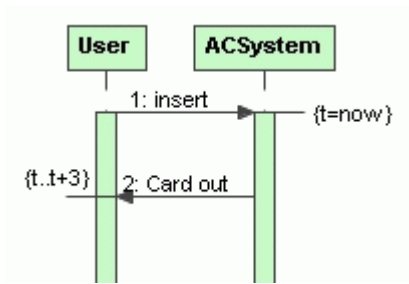
Model elements	Button (hot key)	Notation
Parallel	 (SHIFT+P)	
Break	 (SHIFT+B)	
Negative	 (SHIFT+G)	
Critical Region	 (SHIFT+R)	
Consider	 (SHIFT+C)	
Ignore	 (SHIFT+I)	
Weak Sequencing	 (SHIFT+W)	

Model elements	Button (hot key)	Notation
Strict Sequencing	 (SHIFT+S)	
Assertion	 (SHIFT+R)	
Interaction Use A reference to interactions, communication diagram, sequence diagram, and time diagram can be created.	 (SHIFT+T)	
Message A communication between objects that conveys information with the expectation that an action will ensue. The receipt of a message is one type of event.	 (M)	
Call Message	 (A)	
Send Message	 (E)	

9 UML DIAGRAMS

Sequence Diagram

Model elements	Button (hot key)	Notation
Reply Message	(R)	
Create Message	(C)	
Delete Message	(T)	
Diagonal Message Requires some time to arrive, during which another action occurs.	(D)	
Message to Self	(S)	

Model elements	Button (hot key)	Notation
Recursive message A connected set of messages can be enclosed and marked as iteration.	 (U)	
Duration Constraint A duration defines a value specification that specifies the temporal distance between two time instants.		
Time Constraint Specifies the combination of min and max timing interval values.		

Sequence diagram improvements

State Invariant

Term	The OMG UML specification (UML 2.2: Superstructure) states: "A StateInvariant is a runtime constraint on the participants of the interaction. It may be used to specify a variety of different kinds of constraints, such as values of attributes or variables, internal or external states, and so on".
-------------	---

State invariant element support is added in sequence diagram:

In the Figure 325 on page 672, the State Invariant element is presented in two cases:

1. State Invariant with a assigned mystate State.
2. State Invariant with a defined constraint Y.p=15.

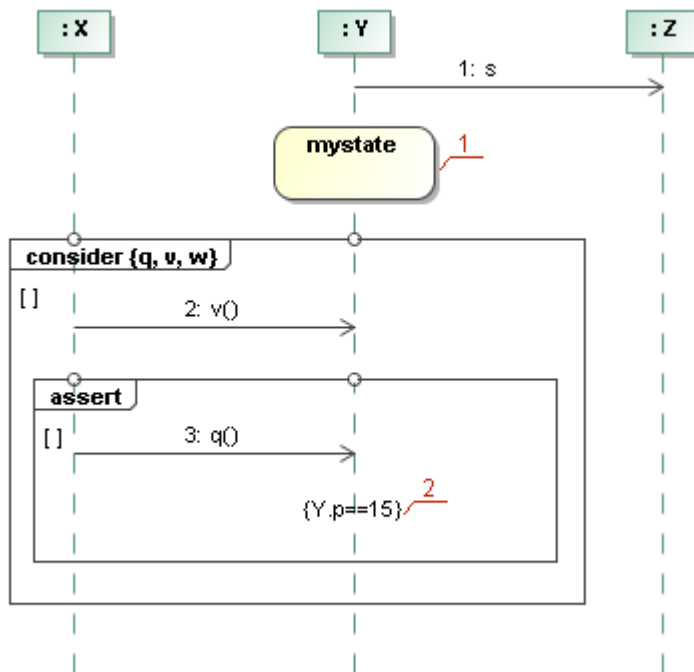


Figure 325 -- State Invariant in a Sequence diagram

Lost / Found Messages

Terms

The OMG UML specification (UML 2.2: Superstructure) states:

"A lost message is a message where the sending event occurrence is known, but there is no receiving event occurrence. We interpret this to be because the message never reached its destination."

"A found message is a message where the receiving event occurrence is known, but there is no (known) sending event occurrence. We interpret this to be because the origin of the message is outside the scope of the description. This may for example be noise or other activity that we do not want to describe in detail."

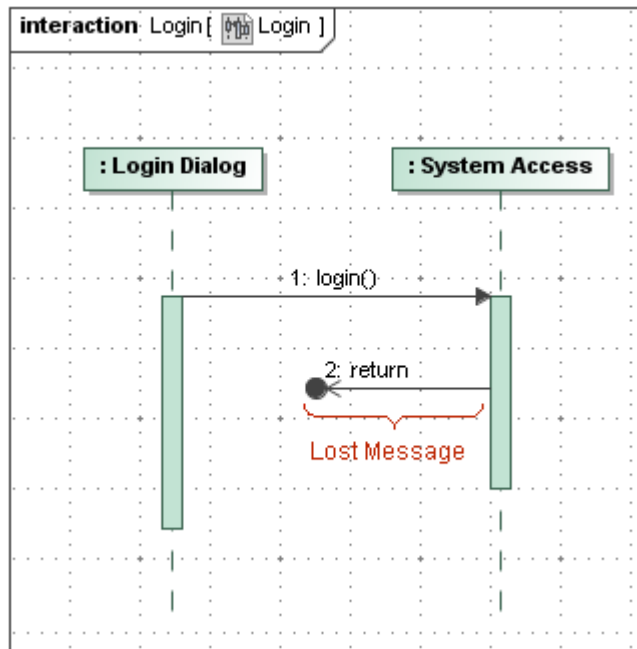


Figure 326 -- Lost Message in a Sequence diagram

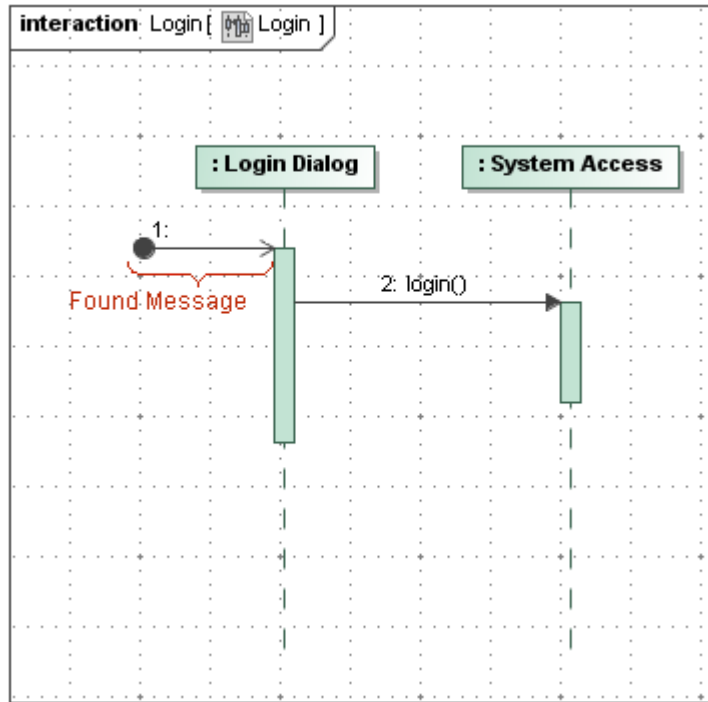


Figure 327 -- Found message in a Sequence diagram

State Machine Diagram

The behavior of objects of a class can be described in terms of states and events, using a state machine connected to the class under construction.

The state machine is a specification of the sequence of states through which an object or an interaction goes in response to events during its life, together with its responsive actions. The state machine may represent the sequence of states of a particular collaboration (i.e. collection of objects) or even the whole system (which is also considered a collaboration). The abstraction of all possible states defined in a state machine is similar to the way class diagrams are abstracted: all possible object types (classes) of a particular system are described.

Objects that do not present a very pronounced reactive behavior may always be considered to stay in the same state. In such a case, their classes do not possess a state machine.

State diagrams (also called Statechart diagrams) represent the behavior of entities capable of dynamic behavior by specifying its response to the receipt of event instances. Typically, the state diagrams describe the behavior of classes, but the statecharts may also describe the behavior of other model entities such as use-cases, actors, subsystems, operations, or methods.

A state diagram is a graph that represents a state machine. States and various other types of vertices (pseudostates) in the state machine graph are rendered by the appropriate state and pseudostate symbols, while transitions are generally rendered by directed arcs that inter-connect them. The states may also contain subdiagrams by physical containment or tiling. Note that every state machine has a top state, which contains all the other elements of the entire state machine. The graphical rendering of this top state is optional.


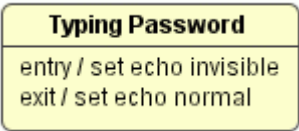

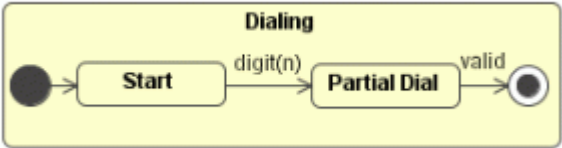

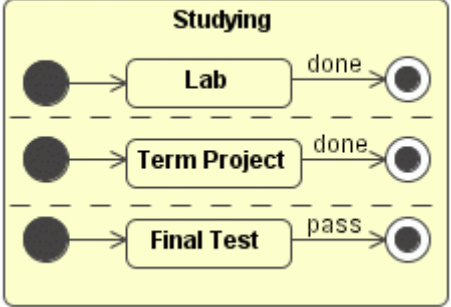
The states are represented by the state symbols, while the transitions are represented by arrows connecting the state symbols.


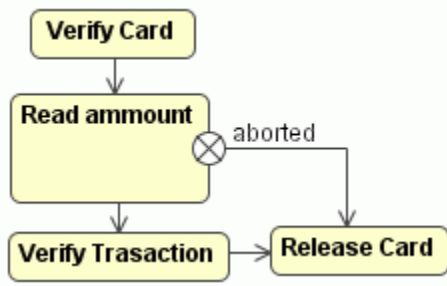








The state diagram concerns with internal object changes (as opposed to the external object interaction in collaboration). Do not attempt to draw them for all classes in the system, because they are used only for modeling a complex behavior. The state diagram shows all the possible states that objects or collaborations may have, and the events that cause the state to change. An event can be another object that sends a message to it: for example, that a specified time has elapsed or that some conditions have been fulfilled. A change of a state is called a transition. A transition may also have an action connected to it that specifies what should be done in connection with the state transition.






State Machine Diagram elements




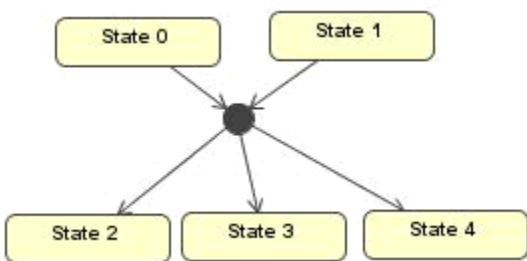

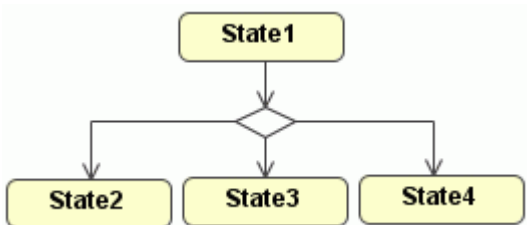
NOTE


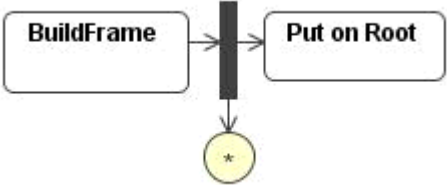


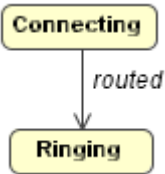


If a black arrow is placed on a button, right-click the button, to open other available buttons.

Model elements	Button (hot key)	Notation
State A state models a situation during which some (usually implicit) invariant conditions holds. The invariant may represent a static situation such as an object waiting for some external events to occur. However, it can also model dynamic conditions such as the process of performing some behavior.s	 (SHIFT+S)	
Composite State A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions.	 (SHIFT+C)	
Orthogonal State An orthogonal state is a composite state with at least 2 regions.	 (C)	

Model elements	Button (hot key)	Notation
Submachine State The submachine state specifies the insertion of the specification of a sub-machine state machine. The Submachine state is a decomposition mechanism that allows factoring of common behaviors and their reuse.	 (A)	
Initial Denotes an initial state when an object is created.	 (I)	
Final State Denotes an object destruction or the end of a collaboration.	 (F)	
Terminate	 (R)	
Entry Point The entry point connection points a reference as the target of a transition. This implies that the target of the transition is the entry point pseudostate as defined in the submachine of the sub-machine state.	 (Y)	

Model elements	Button (hot key)	Notation
Exit Point The exit point connection points a reference as the source of a transition. This implies that the source of the transition is the exit point pseudostate as defined in the submachine of the submachine state that has the exit point connection point defined.	 (U)	
Connection Point Reference The Connection point references of a submachine state can be used as the sources/targets of the transitions. They represent entries into or exits out of the submachine state machine referred by the submachine state.	 (Z)	
Deep History Represents the most recent active configuration of the composite state that directly contains the pseudostate; e.g. the state configuration that was active when the composite state was last exited.	 (P)	

Model elements	Button (hot key)	Notation
Shallow History Represents the most recent active substate of its containing state (but not the substates of that substate). A composite state can have at most one shallow history vertex.	 (SHIFT+R)	
Junction The junction vertices are semantic-free vertices that are used to chain together multiple transitions. They are used to construct the compound transition paths between states.	 (J)	
Choice The choice points are used to split transition paths. In the dynamic choice point, a decision concerning which branch to take is only made after the transition from State1 is taken and the choice point is reached.	 (O)	

Model elements	Button (hot key)	Notation
Fork/Join Vertical Helps to control parallel actions.	 (G)	
Fork/Join Horizontal Helps to control parallel actions.	 (D)	
Transition The starting time for a relative time event may only be omitted for a time event that is the trigger of a state machine.	 (T)	
Transition to Self When an object returns to the same state after the specified event occurs.	 (E)	

Protocol State Machine Diagram

A protocol state machine is always defined in the context of a classifier. It specifies which operations of the classifier can be called, in which state, and under which condition, thus specifying the allowable call sequences on the classifier's operations.





The protocol state machine presents the possible and permitted transitions on the instances of its context classifier, together with the operations that carry the transitions.


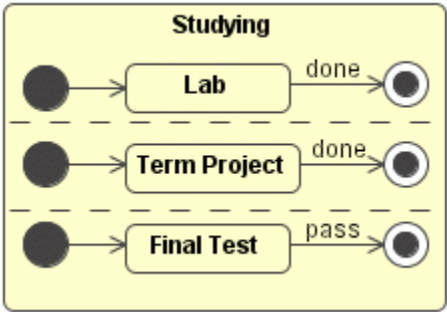

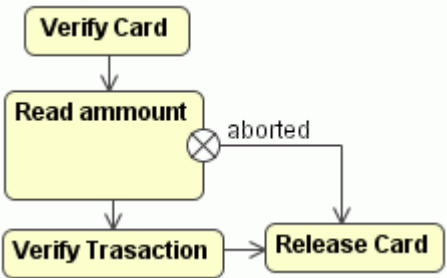






In this manner, an instance lifecycle can be created for a classifier, by specifying the order in which the operations can be activated and the states through which the instance progresses during its existence.






The Protocol State Machine Diagram is created for use with the Protocol State Machine and the Protocol Transitions.


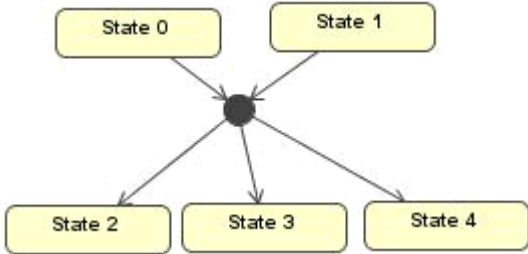

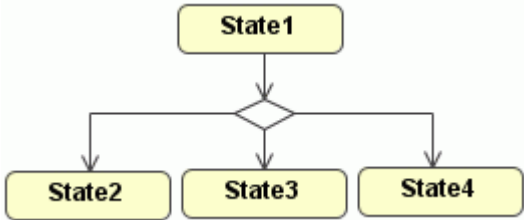

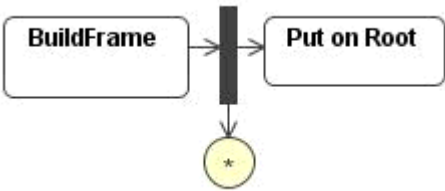

Protocol State Machine Diagram elements


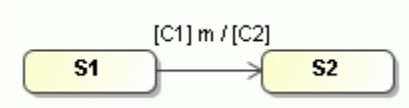

NOTE If a black arrow is placed on a button, right-click the button to open other available buttons.

Model elements	Button (hot key)	Notation
State The states of the protocol state machines are exposed to the users of their context classifiers. A protocol state represents an exposed stable situation of its context classifier: When an instance of the classifier is not processing any operation, the user of this instance can always know its configuration state.	 (SHIFT+S)	
Composite State A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions.	 (SHIFT+C)	

Model elements	Button (hot key)	Notation
Orthogonal State An orthogonal state is a composite state with at least 2 regions.	 (C)	
Submachine State A submachine state specifies the insertion of the specification of a submachine state machine. The submachine state is a decomposition mechanism that allows factoring of common behaviors and their reuse.	 (A)	
Initial Denotes an initial state when an object is created.	 (I)	
Final State Denotes an object destruction or the end of a collaboration.	 (F)	
Terminate	 (R)	

Model elements	Button (hot key)	Notation
Entry Point The entry point connection points a reference as the target of a transition. This implies that the target of the transition is the entry point pseudostate as defined in the submachine of the submachine state.	 (Y)	
Exit Point The exit point connection points a reference as the source of a transition. This implies that the source of the transition is the exit point pseudostate as defined in the submachine of the submachine state that has the exit point connection point defined.	 (U)	
Connection Point Reference The connection point references of a submachine state that can be used as the sources/targets of the transitions. They represent entries into or exits out of the submachine state machine referenced by the submachine state.	 (Z)	

Model elements	Button (hot key)	Notation
Junction The junction vertices are semantic-free vertices that are used to chain together multiple transitions. They are used to construct the compound transition paths between states.	 (J)	
Choice The choice points are used to split transition paths. In the dynamic choice point, a decision on which branch to take is only made after the transition from State1 is taken and the choice point is reached.	 (O)	
Fork/Join Vertical Helps to control parallel actions.	 (G)	
Fork/Join Horizontal Helps to control parallel actions.	 (D)	

Model elements	Button (hot key)	Notation
Protocol Transition A protocol transition (transition as specialized in the ProtocolStateMachines package) specifies a legal transition for an operation. Transitions of the protocol state machines have the following information: a pre condition (guard), on trigger, and a post condition. Every protocol transition is associated to zero or one operation that belongs to the context classifier of the protocol state machine.	 (T)	
Protocol Transition to Self When an object returns to the same state after the specified event occurs.	 (E)	

Activity Diagram






An activity graph is a variation of a state machine. In the state machine, the states represent the performance of actions or subactivities, while the transitions are triggered by the completion of the actions or subactivities. It represents a state machine of a procedure itself. The entire activity diagram is attached (through the model) to a class, such as a use case, or to a package, or to the implementation of an operation. The purpose of this diagram is to focus on flows driven by the internal processing (as opposed to external events). You should use the activity diagrams in situations where all or most of the events represent the completion of internally-generated actions (that is, procedural flow of control). You should use the ordinary state diagrams in situations where asynchronous events occur. An activity diagram is a variant of a state diagram. Organized according to actions, the activity diagrams are mainly targeted towards the representation of the internal behavior of a method (the implementation of an operation) or a use case.



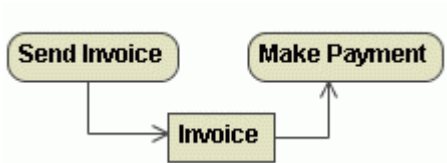

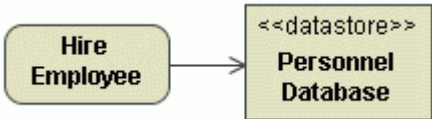
Though activity diagrams are often classified alongside the interaction diagrams, they actually focus on the work performed by a system instead of an object interaction. An activity diagram captures actions and displays their results.

A state diagram may also represent this sequencing of steps. However, given the procedural nature of the implementation of the operations – in which most events simply correspond to the end of the preceding activity – it is not necessary to distinguish states, activities, and events systematically (i.e. state changes and external events have less importance inside the method). It is therefore beneficial to have a simplified representation for directly displaying activities.

The activity diagram provides a convenient way to describe complex algorithms, parallel operations, and business processes. Together with the collaboration and sequence diagrams, they are used to relate use cases.


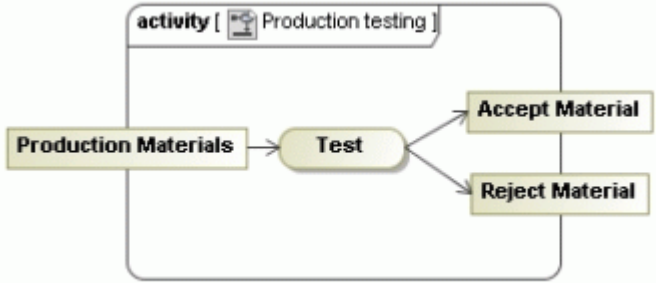


Activity Diagram elements


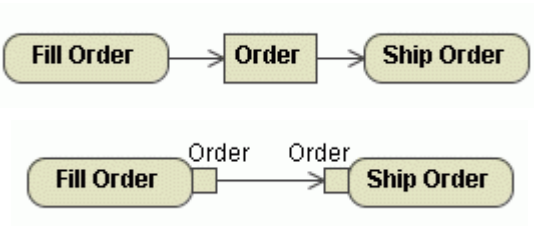





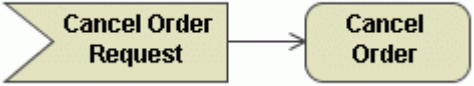
Element	Button (hot key)	Notation
Action An action is a named element that is the fundamental unit of an executable functionality. The execution of an action represents some transformations or processing in the modeled system. When the action is to be executed or what its actual inputs are is determined by the concrete action and the behaviors in which it is used.	 (B)	
Call Operation Action An action that transmits an operation call request to the target object, where it may cause the invocation of the associated behavior. The argument values of the action are available to the execution of the invoked behavior.	 (O)	
Opaque Action An opaque action is introduced for implementing specific actions or for use as a temporary placeholder before some other actions are chosen.		

Element	Button (hot key)	Notation
Any Action This element is introduced in order to maintain any other desirable action element with an appropriate metaclass stereotype applied.		
Object Node The Activity nodes are introduced to provide a general class for the nodes connected by activity edges. The ActivityNode replaces the use of the StateVertex and its children for activity modeling in UML.	 (SHIFT+B)	
Data Store A data store node is a central buffer node for a non-transient information. A data store keeps all tokens that enter it, copies them when they are chosen to move downstream. Incoming tokens containing a particular object replace any tokens in the object node containing that object.	 (SHIFT+D)	

9 UML DIAGRAMS






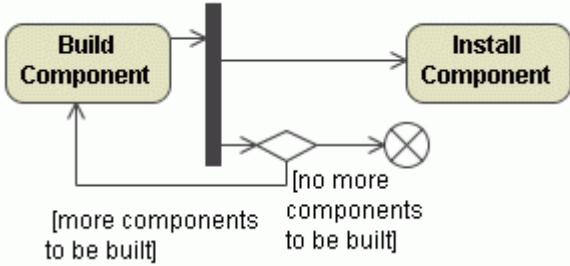

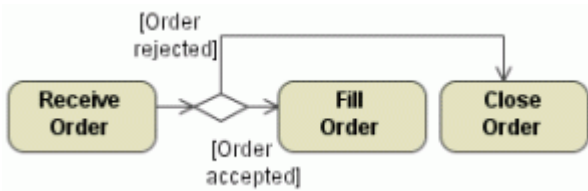
Activity Diagram

Element	Button (hot key)	Notation
Activity Parameter Node It is an object node for inputs and outputs to the activities. The Activity parameters are object nodes at the beginning and end of the flows, to accept inputs to an activity and provide outputs from it.		
Input Expansion Node An expansion node is an object node used to indicate a flow across the boundary of an expansion region. A flow into a region contains a collection that is broken into its individual elements inside the region, which is executed once per element.		
Output Expansion Node A flow out of a region combines individual elements into a collection for use outside the region.		

Element	Button (hot key)	Notation
Object Flow Is an activity edge that can have objects or data passing along it. An object flow models the flow of values to or from the object nodes.	 (SHIFT+F)	
Control Flow Is an edge that starts an activity node after the previous one is finished. Objects and data cannot pass along the control flow edge.	 (F)	
Send Signal Action Is an action that creates a signal instance from its inputs, and transmits it to the target object, where it may trigger the state machine transition or the execution of an activity.	 (SHIFT+S)	
Accept Event Action Is an action that waits for the occurrence of an event that meets the specified conditions. The Accept event actions handle event occurrences detected by the object owning the behavior.	 (E)	


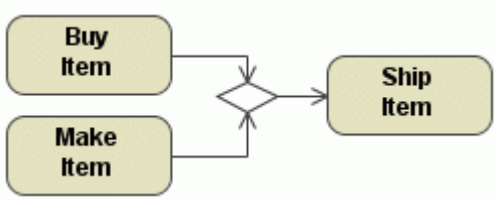

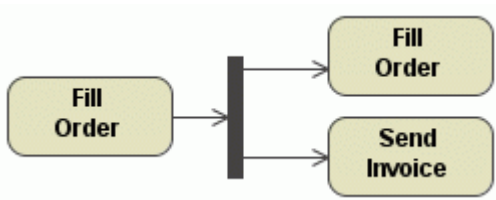

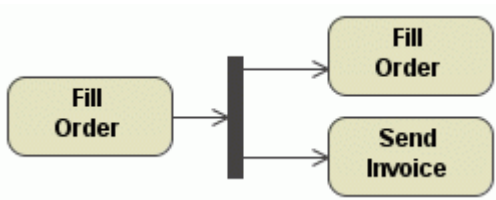

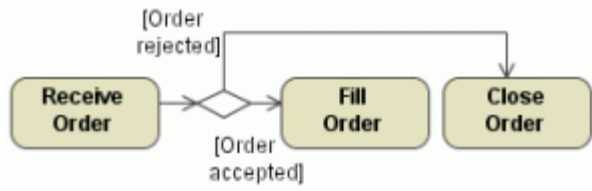
9 UML DIAGRAMS







Activity Diagram

Element	Button (hot key)	Notation
Initial Node An initial node is a starting point for executing an activity. It has no incoming edges.	 (T)	
Activity Final An activity final node is a final node that stops all flows in an activity.	 (D)	
Flow Final The Final node that terminates a flow and destroys all tokens that arrive at it. It has no impact on other flows in the activity.	 (L)	
Decision Decision is a control node that chooses between outgoing flows. A decision node has one incoming edge and multiple outgoing activity edges.	 (G)	

9 UML DIAGRAMS




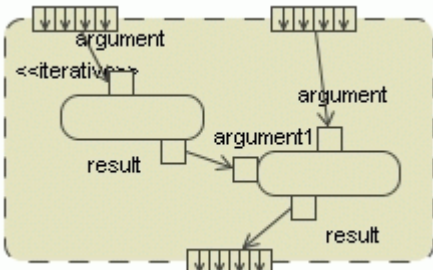


Activity Diagram

Element	Button (hot key)	Notation
Merge A merge node is a control node that brings together multiple alternate flows. It is not used to synchronize concurrent flows but it is used to accept one among several alternate flows.	 (G)	
Fork/Join Horizontal Helps to control parallel actions.	 (K)	
Fork/Join Vertical Helps to control parallel actions.	 (SHIFT+K)	
Exception Handler An exception handler is an element that specifies a body to execute in case the specified exception occurs during the execution of the protected node.	 (P)	

Element	Button (hot key)	Notation
Structured Activity Node A structured activity node is an executable activity node that may have an expansion into the subordinate nodes. The structured activity node represents a structured portion of the activity that is not shared with any other structured node, except for nesting.		
Conditional Node A conditional node is a structured activity node that represents an exclusive choice among alternatives.		
Sequence Node A sequence node is a structured activity node that executes its actions in order.		


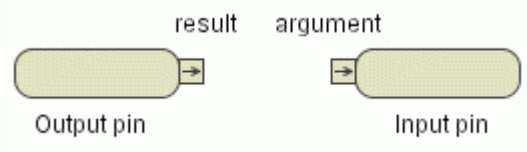





9 UML DIAGRAMS

Activity Diagram

Element	Button (hot key)	Notation
Loop Node A loop node is a structured activity node that represents a loop with the setup, test, and body sections.		
Expansion Region An expansion region is a structured activity region that executes multiple times corresponding to the elements of an input collection.		
Time Event A time event specifies a point of time by an expression. The expression might be absolute or might be relative to some other points of time.		

9 UML DIAGRAMS

Activity Diagram

Element	Button (hot key)	Notation
Input Pin An input pin is a pin that holds input values to be consumed by an action. They are object nodes that receive values from other actions through object flows.	 (SHIFT+I)	
Output Pin A pin that holds output values produced by an action. Output pins are object nodes that deliver values to other actions through object flows.	 (SHIFT+O)	
Value Pin A value pin is an input pin that provides a value to an action that does not come from an incoming object flow edge.		
Swimlanes Swimlanes are used to organize responsibility for actions and subactivities according to the class, dividing an activity diagram.	 (SHIFT+V)  (SHIFT+W)	

Smart Activity Diagram layout

Dynamic centerlines

The centerlines are displayed only when a center of the shape that was moved or newly drawn is located near the center of another shape that already exists in the diagram. These lines help to draw diagram with aligned shapes easily.

When the center of the shape that was moved coincides with a center of any shape that is placed to its right or left, a horizontal centerline is displayed. When the center of the shape is close to any center of a shape that is located above or below it, a vertical centerline is displayed.

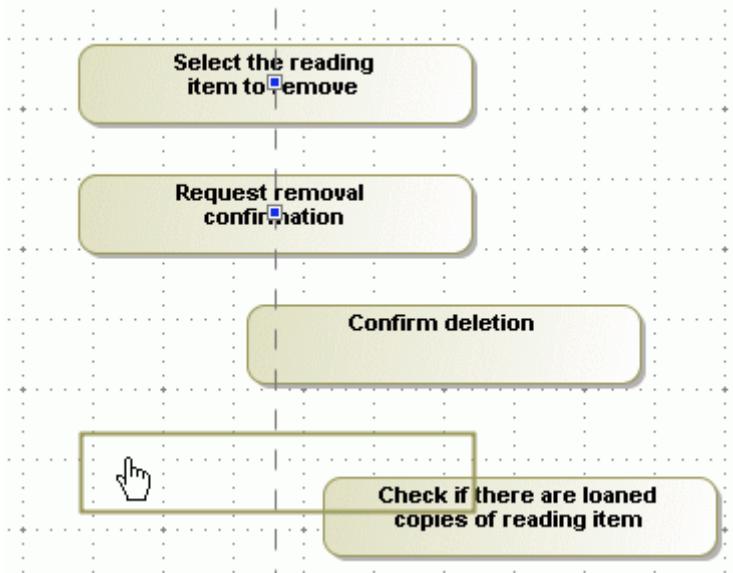


Figure 328 -- Dynamic vertical centerline

To switch off the dynamic centerlines


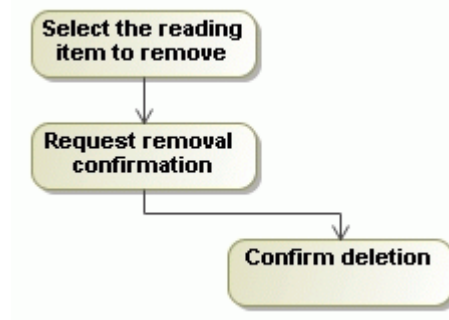
- Click the **Show Centerlines** button  in the diagram toolbar or press **C**.
- From the **Options** main menu, select **Environment**. In the open dialog box, click the **Diagram** node and clear the **Show centerlines in flow diagrams** check box in the **Display** properties group.

Diagram orientation

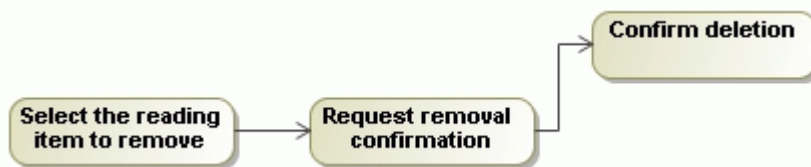
The diagram orientation is used to assign the correct rectilinear path breaks and draw paths between the activity diagram shapes. The paths can be drawn from side to side, or from the lower to the upper shape borders.

Example:

For a vertical diagram orientation - in this case if two shapes are not in the same centerline, the paths will be connected from the lower border of the first shape to the upper border of the next shape, adding break points:



For a horizontal diagram orientation - in this case the paths will be connected from the side border of the first shape to the next side border of the second shape, adding break points:



To change the diagram orientation

- From the **Options** main menu, select **Project**. In the open dialog box, expand **Diagram** group and in the right side properties list, change the value for the **Diagram Orientation** property.
- From the diagram pane shortcut menu, select **Diagram Properties** and change the value for the **Diagram Orientation** property.

For applying the Activity diagram layout tool, see “Activity Diagram Layout Tool” on page 305.

Implementation Diagram

The Implementation diagrams help developers describe the logical software structure inside a computer system or across a large distributed system. They show aspects of the physical implementation, including the structure of components and the run-time deployment system.

They come in two forms:

1. **Component diagrams** that show the structure of the components, including the classifiers that specify them and the artifacts that implement them; and
2. **Deployment diagrams** that show the structure of the nodes on which the components are deployed.

These diagrams can also be applied in a broader way on the business modeling where the components represent the business procedures and artifacts, and the deployment nodes represent the organization units and resources (human and otherwise) of the business.

Component diagram overview

A component diagram represents a physical structure of a code (as opposed to the class diagram, which portrays the logical structure) in terms of code components and their relationships within the implementation environment. A component can be a source code component, a binary component, or an executable component.

A component contains information about the logical class or classes that it implements, thus creating a mapping from a logical view to a component view. Dependencies between the components are shown, making it easy to analyze how a change in one component affects the others. The components may also be shown with any of the interfaces that they expose. They, as with almost any other model elements, can be grouped into packages, much like classes or use cases.








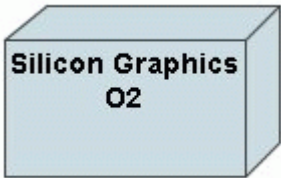
The component diagrams are used in the later phases of the software development, when there is a need to divide up classes among different components. When working with the CASE facilities, the components are used for file-class mapping during code generation, reverse engineering, and round-trip engineering operations.


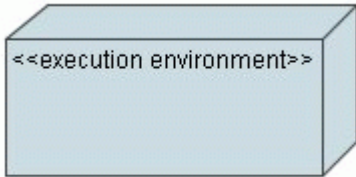

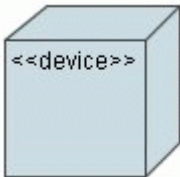

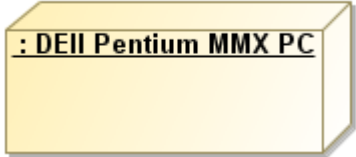

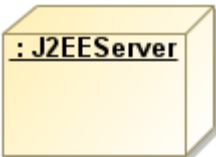
Deployment diagram overview


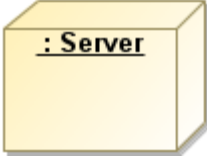






The Deployment diagrams show the physical layout of the various hardware components (nodes) that compose a system as well as the distribution of executable programs (software components) on this hardware.

The Deployment diagrams are crucial when dealing with distributed systems. You may show the actual computers and devices (nodes), along with the connections they have to each other, thus specifying a system topology. Inside the nodes, the executable components and objects are located in a way that it shows where the software units are residing and on which nodes they are executed. You may also show dependencies between the components.

Implementation Diagram elements








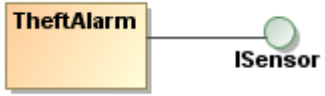

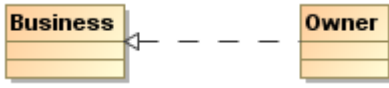
Element	Button (hot key)	Notation
Component The Component may be used to define the requirements for each physical software element.	 (K)	
Component Instance An instance of a component.	 (A)	
Port Ports represent interaction points between a classifier and its environment. A port has the ability to specify that any requests arriving at this port are handled.	 (SHIFT+R)	
Node A Node is a physical object that represents a processing resource, generally having at least a memory and often the processing capability as well.	 (O)	


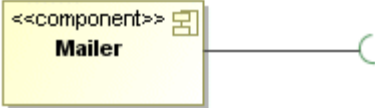

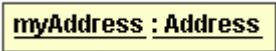
Element	Button (hot key)	Notation
Execution Environment The element that is used to indicate that a node is the execution environment.		
Device A physical computational resource with the processing capability upon which artifacts may be deployed for an execution.		
Node Instance An instance of a node.	 (T)	
Execution Environment Instance The element that is used to indicate that a node is an instance of the execution environment.		


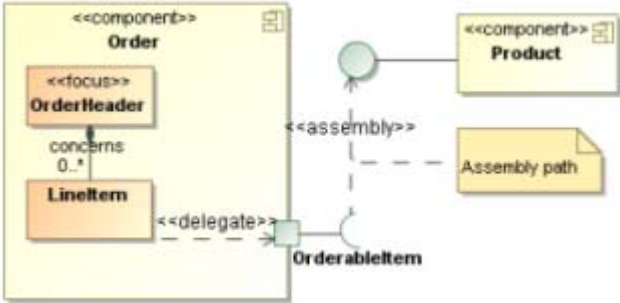

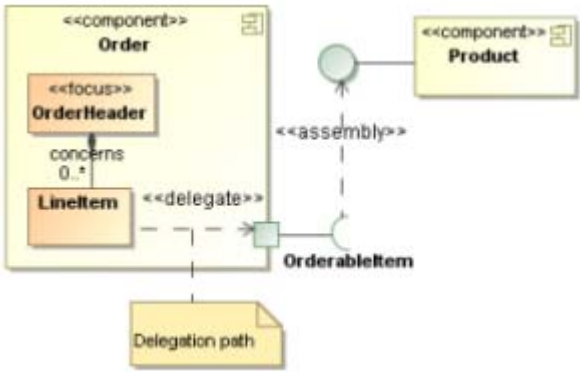
Element	Button (hot key)	Notation
Device Instance		
Package A group of classes and other model elements.	 (P)	
Artifact An artifact represents a physical piece of information that is used or produced by a software development process. Examples of Artifacts include models, source files, scripts, and binary executable files. An Artifact may constitute the implementation of a deployable component.	 (B)	
Deployment Specification It indicates a set of properties that defines how a component should be deployed.		


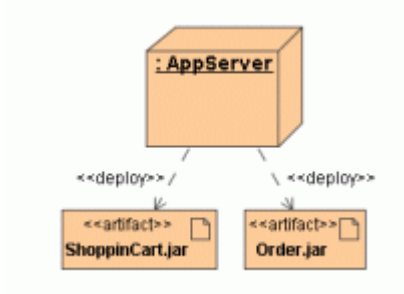

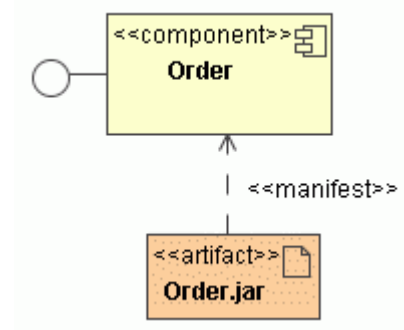
9 UML DIAGRAMS


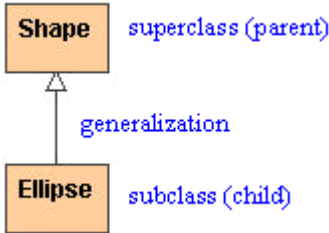



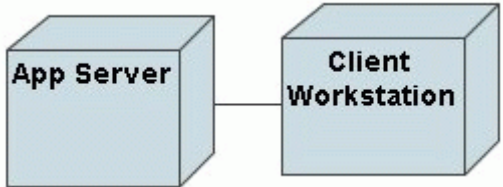
Implementation Diagram

Element	Button (hot key)	Notation
Artifact Instance		
Deployment Specification Instance		
Interface All functionalities implemented by a particular component.	 (F)	
Interface Realization The classifier at the tail of the arrow implements the interface that is located at the arrow head or uses that interface. Note: Choose a different Interface Realization direction from the toolbar to draw a line with an opposite arrow end.	 (R)	
Realization A realization signifies that the client set of elements are an implementation of the supplier set, which serves as the specification.	 (E)	

Element	Button (hot key)	Notation
<p>Usage</p> <p>A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation.</p> <p>Usage is also used to create required interface.</p> <p>Note: Choose a different Usage direction from the toolbar to draw a line with an opposite arrow end.</p>	 <p>(U)</p>	
<p>Instance</p> <p>An instance specifies the existence of an entity in a modeled system and completely or partially describes the entity. It has a unique name, state, and classifier as the type of the identifier.</p>	 <p>(SHIFT+O)</p>	

Element	Button (hot key)	Notation
<p>Assembly Connector</p> <p>An assembly connector is a connector between two components that defines that one component provides the services that another component requires.</p>		
<p>Delegation Connector</p> <p>A delegation connector is a connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component's parts.</p>		






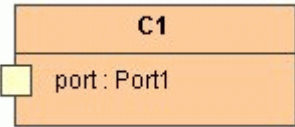

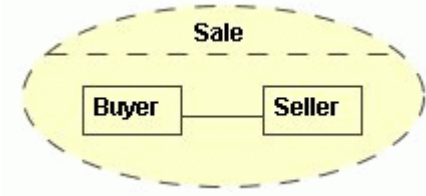
Element	Button (hot key)	Notation
Deployment A deployment is the allocation of a deployment target to an artifact or artifact instance.		
Manifestation A manifestation is the concrete physical rendering of one or more model elements by an artifact.		






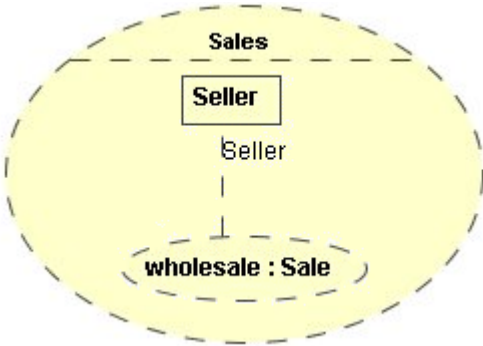
Element	Button (hot key)	Notation
Generalization A relationship between a specification element and an implementation element.	 (G)	
Link A relationship between a specification element and an implementation element.	 (L)	
Communication Path A communication path is an association between two DeploymentTargets, through which they are able to exchange signals and messages.		

Composite Structure Diagram

The Composite Structure diagram allows a decomposition and modeling of the internal structure of the classifiers.

Composite Structure Diagram elements

Element	Button (hot key)	Notation
Classifier Is extended with the capability to own collaboration uses. These collaboration uses link a collaboration with the classifier to give a description of the works of the classifier.	 (SHIFT+G)	
Part	 (P)	
Port A port may appear either on a contained part representing a port on that part, or on the boundary of the class diagram, representing a port on the represented classifier itself.	 (SHIFT+R)	
Collaboration A collaboration describes a structure of collaborating elements (roles), each performing a specialized function, which collectively accomplishes some desired functionalities. Its primary purpose is to explain how a system works and, therefore, it typically only incorporates those aspects of reality that are deemed relevant to the explanation.	 (Q)	

Element	Button (hot key)	Notation
Collaboration Use A collaboration use represents one particular use of a collaboration to explain the relationships between the properties of a classifier. The collaboration use shows how the pattern described by a collaboration is applied in a given context, by binding specific entities from that context to the roles of the collaboration.	 (U)	
Connector Each connector may be attached to two or more connectable elements, each representing a set of instances. Each connector end is distinct in the sense that it plays a distinct role in the communication realized over the connector.	 (C)	
Role Binding Is a relationship from parts to the Collaboration Use (in diagram).	 (B)	

Displaying existing Parts on the Composite Structure diagram creation

On a new Composite Structure diagram creation, if context classifier has properties, the **Select Part** dialog will open. Select Parts to be drawn automatically in the Composite Structure diagram.

Case study to display existing parts on the Composite Structure diagram [Example is taken from UML 2 Superstructure Specification]:

1. Create the *Observer* package with *CallQueue*, *SlidingBarIcon* classes inside and *Observer* collaboration.
2. To the *Observer* collaboration add *Subject* property (*CallQueue* type) and *Observer* property (*SlidingBarIcon* type) (see Figure 329 on page 710).
3. Create a composite structure diagram inside the collaboration. The **Select Parts** dialog will open (see Figure 330 on page 711).
4. Select properties (parts) to display them in a diagram. Click OK.
5. Parts are displayed in a diagram (Figure 331 on page 711).

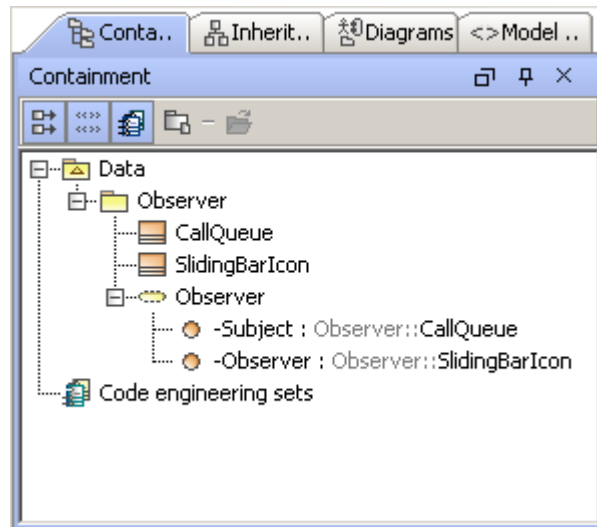


Figure 329 -- Sample with Observer Collaboration

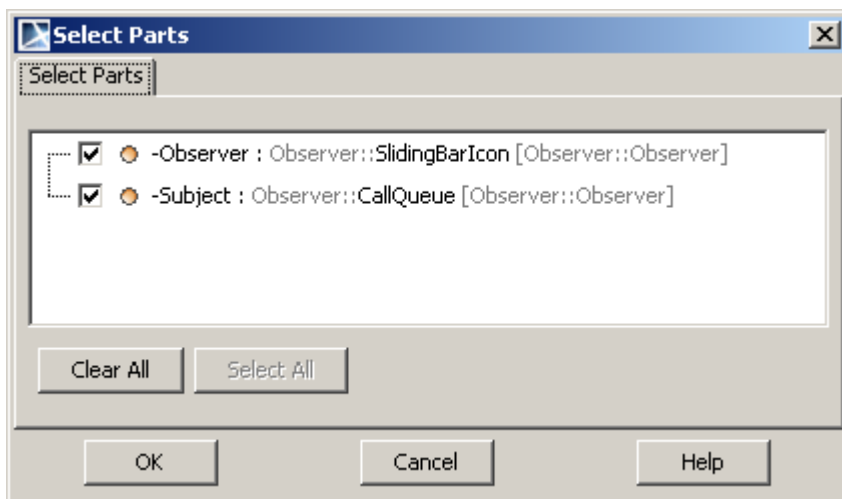


Figure 330 -- The Select Parts dialog box

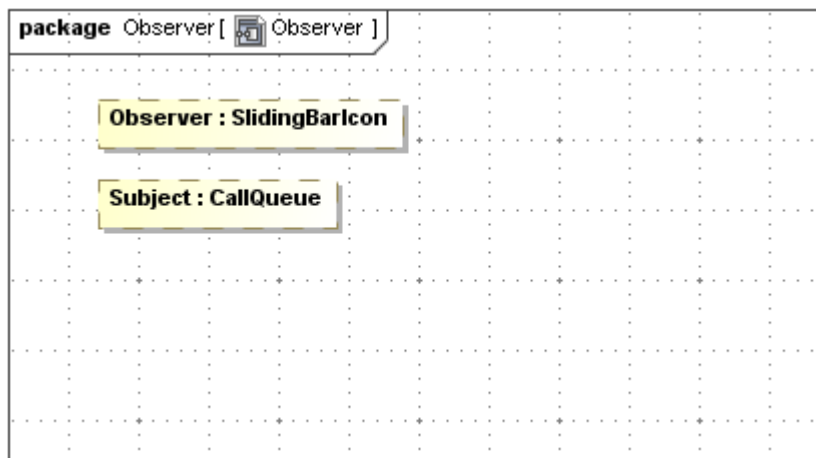


Figure 331 -- Observer composite structure diagram with parts

Interaction Overview Diagram

The Interaction Overview diagram focuses on the overview of the flow of control between Interactions. It is based on the Activity diagram notation. Interactions in the Interaction Use diagram are represented using the Interaction Use element. See the sample of the Interaction Overview diagram in Figure 332 on page 713.

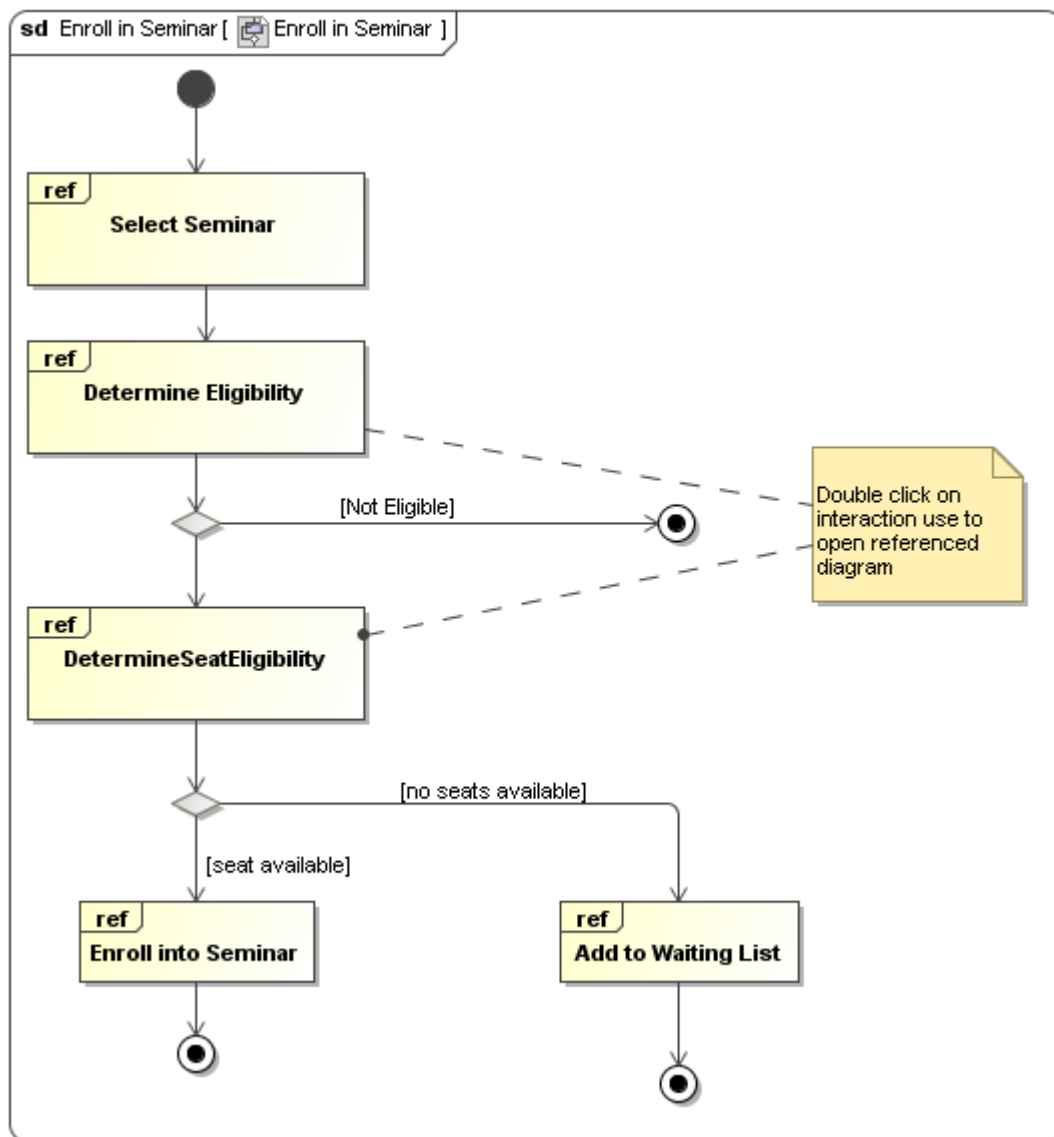




Figure 332 -- Sample of the Interaction Overview diagram

Element	Button (hot key)	Notation
Interaction Use Represents interactions.	 (Shift-T)	

Other elements of the Interaction Overview diagram are the same as in the Activity diagram. For more information about the Activity diagram, see “Activity Diagram elements” on page 687.

10 EXTENSION DIAGRAMS

General Information

MagicDraw supports an extension to UML diagrams:

- "Content Diagram" (Standard, Professional, Architect, and Enterprise editions only)
- "Robustness Diagram" (Standard, Professional, Architect, and Enterprise editions only)
- "User Interface Modeling Diagram" (Standard, Professional, Architect, and Enterprise editions only)

Also, MagicDraw provides various types of other diagrams:

- "Web Diagram" (Standard, Professional, Architect, and Enterprise editions only)
- "CORBA IDL Diagram" (Architect and Enterprise editions only)
- "Generic DDL Diagram" (Architect and Enterprise editions only)
- "Oracle DDL Diagram" (Architect and Enterprise editions only)
- "WSDL Diagram" (Architect and Enterprise editions only)
- "XML Schema Diagram" (Architect and Enterprise editions only)
- "Time Diagram" (Standard, Professional, Architect, and Enterprise editions only)
- "Struts Diagram" (Professional, Architect, and Enterprise editions only)
- "Networking Diagram" (Standard, Professional, Architect, and Enterprise editions only)
- "Business Process Diagram" (Standard, Professional, Architect and Enterprise editions only)

Working with Diagrams

For a general information about working with diagrams, see Chapters "Diagrams Basics" and "UML Diagrams".

In the **Environment Options** dialog box, **Plugins** pane, you may select what extension diagrams you want to enable or disable.

Patterns

Various types of classes can be created in every class diagram using a **Pattern Wizard**. It contains GOF, Java, Junit, CORBA IDL, XML Schema, and WSDL design patterns.








New patterns and modifications of the existing ones can also be created using Java code or JPython scripts. For a detailed description, see [MagicDraw Open Api user's guide](#).








For a detailed description of the **Pattern Wizard**, see “Controlling Merge memory usage” on page 465.

To open the Pattern Wizard

- In the class diagram, click the **Class by Pattern** button.
- From the class shortcut menu, select **Tools** and then select the **Apply Pattern** subcommand.
- Select a class and select **Apply Pattern** from the **Tools** main menu.

Common Elements

Toolbar Button	Button (Hot key)	
Selection	 (Escape)	
Sticky Button	 (Z)	
Text Box	 (X)	
Text Box (HTML Text)	 (SHIFT+X)	
Note	 (N)	
Note (HTML Text)	 (SHIFT+N)	
Comment		

Toolbar Button	Button (Hot key)	
Comment (HTML Text)		
Anchor	 (H)	
Constraint	 (SHIFT+H)	
Containment	 (SHIFT+C)	
Dependency	 (SHIFT+D)	
Separator	 (W)	
Rectangular Shape	 (SHIFT+Q)	



User Interface Modeling Diagram

Overview

User Interface (UI) Modeling diagram make it possible to build prototypes of user interfaces, using MagicDraw, connect UI mock-ups with whole Architectural model, export them as images, and create browsable reports for presentations. In short, they help gather information faster and thus save time and money.

The merits of User Interface Modeling lies in its ability to:

- Rapidly create WYSIWYG User Interface prototypes.
- Integrate User Interface development with UML specifications.
- Get immediate feedback from prospective users on real situations and reuse it for next designs.
- Create browsable reports with the **MagicDraw Report Wizard** (for more information about report generation see "MagicDraw ReportWizard UserGuide.pdf").

Why Prototyping User Interface Modeling?

Various versions of user interfaces need to be tested in order to see how clients respond. This is especially true if you work on key dialogs like for example a sign up screen for a website or an e-commerce application. Working with User Interface prototypes instead of "real" user interfaces will thus enable you to work with small details such as color and the position of a button and substantially reduce designer and programmer overhead in the design phase.

Build up mock-ups or prototypes, get quick feedback from prospective users, and take and reuse the feedback for future designs. All this is possible with User Interface prototyping. The feedback loop makes for quicker mature designs that work for everybody, which is what really matters.

You can build mock-ups or prototypes to meet following objectives:

- **Integrate GUI development with UML specifications** since this is a new field with increased design capability and since it is easier to see missing parts in UML or User Interface modeling.

- **Help business analysts gather requirements** because of the permanent feedback from prospective users, which makes it easier for them to get all the information needed.
- **Create browsable User Interface prototypes / GUI simulation / Story boarding** since several prototypes can be connected via hyperlinks with one another and then be presented together in a report and simulate the workflow of an application.
- **Resolve flow issues** as it is easier to think through a problem when a user-related interface can be changed quickly.
- **Get "buy-in" from stakeholders** as it can be shown more rapidly how a particular user interface will look like.
- **Run a usability test before full production** so that potential errors in usability like overfilled screens or a usage too complicated can be avoided.
- **Test a series of interaction widgets.** You may think, for example, that another text field or button would be good on a screen. Since modifying a prototype is so easy, it is not a problem to present these suggestions to others.

For more information about prototyping, go to:

<http://today.java.net/pub/a/today/2005/08/23/prototyping.html>


<http://www.scottberkun.com/essays/12-the-art-of-ui-prototyping/>

Working with User Interface Modeling Diagrams

Starting Working with a User Interface Modeling Diagram

To starting working with a User Interface Modeling Diagram:

Either:

- Click the **User Interface Diagram**  button in the Diagram toolbar.
- Or select **New Diagram > Custom Diagrams > User Interface Diagram** from the Package or Model shortcut menu in the Browser.

Creating a User Interface Model

After a project is loaded and a diagram created, User Interface components can be added to the surface.

To add User Interface components:

- Simply drag and drop them out of the diagram's toolbar.

For further information, see “User Interface Modeling” on page 721 and “Case Studies for User Interface Modeling” on page 740.

Loading a Sample with an Already Created User Interface Model

To load a sample with an already created User Interface model:

1. Open the **UI Modeling Samples.mdzip** or **UI Modeling in System Development.mdzip** projects which you will find in the <MagicDraw installation folder>/samples/diagrams/User Interface Modeling directory.
2. After loading the sample, select the models from the **Index-Diagram** by clicking the hyperlinks.

User Interface Modeling

To model a user interface:

1. Click the button for one of the graphical components in the diagram toolbar and drop it on the diagram surface. All the components mentioned in Table 5, “User Interface Modeling Components,” on page 723, “User Interface Modeling Components” on page 722 will then be drawn.
2. Specify then every component by editing the element properties. See “Specifying Elements” on page 734.

For icon usage, go to “Icon Usage” on page 737.

For how to work with symbol properties, see “Using Symbol Properties” on page 738.

Graphical components can be nested in each other, see “Nesting” on page 733. They can also be reused in other models, see “Reusability” on page 734.








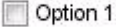
For a short introduction on User Interface prototyping, see “User Interface Prototyping” on page 739.




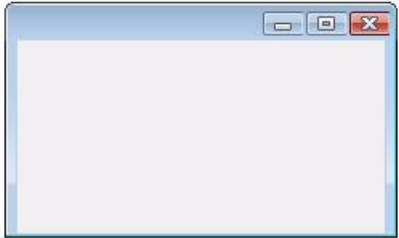

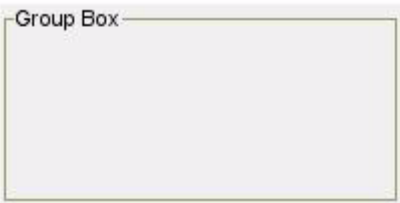


For User Interface Modeling, see “Case Studies for User Interface Modeling” on page 740. Three case studies will provide step-by-step instructions as to how to build user interfaces and create browsable reports.


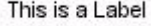

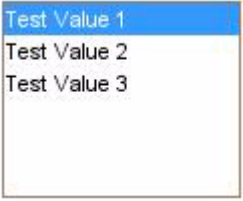




User Interface Modeling Components






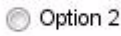


Table 5, “User Interface Modeling Components,” on page 723 lists all available User Interface modeling components and their properties:







TABLE 5. User Interface Modeling Components




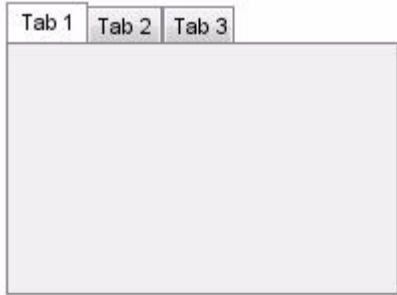

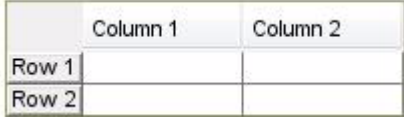


Component	Description	Properties	Screenshot
 Button	A regular button with the possibility to add text, icon or both.	<ul style="list-style-type: none"> - Icon: select one of the available icons or choose your own. - Inactive: activates/deactivates the button. - Text: displays text on the button on the diagram pane. - Selected: sets the button as selected or not. 	
 Toolbar button with Icon	Small sized button with pre-defined icon (close, copy, cut, delete, new, open, paste, print, redo, save, search, undo, zoom in, zoom out).	- same as for regular buttons.	
 Button with Text	Button with pre-defined text (Back, Cancel, Close, Finish, Next, OK).	- same as for regular buttons.	
 Check Box	Check box with text.	<ul style="list-style-type: none"> - Inactive: activates/deactivates the box. - Text: displays text next to the check box on the diagram pane. - Selected: marks/unmarks the check box. 	






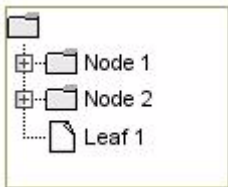
Component	Description	Properties	Screenshot
 Combo Box	Regular combo box in non-editable style with possibility to show a value.	<ul style="list-style-type: none"> - Inactive: activates/deactivates the box. - Text: displays text in the combo box. 	
 Frame*	Main container component represented by a regular Internal Frame. Any other component can be nested inside.	<ul style="list-style-type: none"> - Icon: select one of the available icons or choose your own. - Inactive: activates/deactivates the frame. - Maximize: defines if the maximize-icon should be visible or hidden. - Minimize: defines if the minimize-icon should be visible or hidden. - Title: displays the title of the frame. 	
 Group Box*	A panel with a titled border. Any other component can be nested inside.	<ul style="list-style-type: none"> - Border Type: defines the style of the border. - Title: displays the title of the border. - Titled: defines whether border text should be shown or not. 	
 Hyperlink	A blue colored and underlined item for showing some text or an icon or both.	<ul style="list-style-type: none"> - Icon: select one of the available icons or choose your own. - Inactive: activates/deactivates the link. - Text: displays the text of the hyperlink. 	

Component	Description	Properties	Screenshot
 Label	Item with ability to present a text, an icon or both.	<ul style="list-style-type: none"> - Icon: select one of available icons or choose your own. - Inactive: activates/deactivates the label. - Text: displays the text in the label. 	
 List	Bordered item which stores and shows numerous values. Values can be shown as selected.	<ul style="list-style-type: none"> - Horizontal Scroll Bar: defines the visibility of the horizontal bar. - Inactive: activates/deactivates the list. - Selected Value: defines which value should be shown as selected. - Values: add/remove values to/from the list. - Vertical Scroll Bar: defines the visibility of the vertical bar. 	
 Menu Bar	A bar that can present several text items.	<ul style="list-style-type: none"> - Values: add/remove menus to/from the bar. 	
 Panel*	A bordered panel.		

Component	Description	Properties	Screenshot
 Password Field	A bordered field with a text which is hidden through symbols.	<ul style="list-style-type: none"> - Hidden: decides whether the text should be shown as text or symbolized. - Inactive: activates/deactivates the field. - Text: displays the text of the password field. 	
 Progress Bar	A bar which presents the state of progress.	<ul style="list-style-type: none"> - Maximum Value: sets the maximum value of the bar. - Minimum Value: sets the minimum value of the bar. - Value: sets the display text on the bar. - Vertical: switches between horizontal and vertical orientation. 	
 Radio Button	A radio button with possibility to present some text.	<ul style="list-style-type: none"> - Inactive: activates/deactivates the button. - Text: displays text next to the radio button. - Selected: marks/unmarks the button. 	
 Scroll Bar	Item which represents regular scroll bar.	<ul style="list-style-type: none"> - Inactive: activates/deactivates the bar. - Vertical: switches between horizontal and vertical orientation. 	

Component	Description	Properties	Screenshot
 Scroll Pane*	A panel which can contain a vertical and/or a horizontal scroll bar.	<ul style="list-style-type: none"> - Horizontal Scroll Bar: defines the visibility of the horizontal bar. - Vertical Scroll Bar: defines the visibility of the vertical bar. 	
 Separator	A line with ability to split up a component.	<ul style="list-style-type: none"> - Vertical: switches between horizontal and vertical orientation. 	
 Slider	An item for presenting a range of several values. The knob of this item can be moved to these values.	<ul style="list-style-type: none"> - Inactive: activates/deactivates the slider. - Invert: switches maximum and minimum values. - Knob Position: defines the location of the knob. - Maximum Value: sets the maximum value of the slider. - Minimum Value: sets the minimum value of the slider. - Spacing: sets the space between two markings. - Values: defines at which position the slider should be a text. - Vertical: switches between horizontal and vertical orientation. 	

Component	Description	Properties	Screenshot
 Spinner	Regular spinner that can show a value.	<ul style="list-style-type: none"> - Inactive: activates/deactivates the spinner. - Text: displays value on the spinner. 	
 Tabbed Pane*	Item which represents a panel with tabs.	<ul style="list-style-type: none"> - Active Tab: sets which tab should be shown as selected. - Tab Position: selects the placement of the tabs. - Tabs: add/remove tabs to/from the pane. 	
 Table	A table item with the ability to add and/or remove columns and/or rows. See section "Modifying a Table" on page 729 for more information.	<ul style="list-style-type: none"> - Column Header: defines whether the column header should be visible or not. - Row Header: defines whether the row header should be visible or not. 	
 Text Area	A multi-line bordered item to present long text.	<ul style="list-style-type: none"> - Horizontal Scroll Bar: defines the visibility of the horizontal bar. - Inactive: activates/deactivates the area. - Text: displays text in the text area. - Vertical Scroll Bar: defines the visibility of the vertical bar. 	

Component	Description	Properties	Screenshot
 Text Field	A single-line bordered item to present some text.	<ul style="list-style-type: none"> - Inactive: activates/deactivates the field. - Text: displays text in the text field. 	
 Tool Bar*	A bar which can consist of any other component. Usually used to nest buttons or labels with icons.		
 Tree	An item to present a tree structure. It can add numerous nodes and/or leaves to it. For information on how to do this, see section "Modifying a Tree" on page 733.	<ul style="list-style-type: none"> - Expand: defines whether the whole tree should be expanded or not. - Horizontal Scroll Bar: defines the visibility of the horizontal bar. - Icon: select one of available icons or choose your own. - Text: displays the text of the root node. - Vertical Scroll Bar: defines the visibility of the vertical bar. 	

*** NOTE**

The components marked with a * are **Container** elements and can nest nest other components in them.

Modifying a Table

You can add columns and rows to a table, order and name them. When expanding a table in the Browser tree, the structure of the table will be displayed. Rows are added directly to the model element of the table. Columns are stored in the Columns Enumeration. For every new column a new Cell

will automatically be added to every row. To add a value to the table, you need to edit the name of the Cell.

You can edit a table in the following ways:

- Add a row to the table.
- Change the order of the rows.
- Add a column to the table.
- Change the order of the columns.
- Add a value to the table.

Adding a row

To add a row to a table:

1. Right-click on the table element in the Browser tree to open the shortcut menu.
2. Select **New Element** and then **Row**. New row will be added to the table.
3. Type in a title for the new row.

Changing the row order

To change the order of the rows:

1. Select the table in the diagram.
2. Right-click on it and select **Specification** in the shortcut menu.
3. Go to the Rows node - all rows are listed there.
4. Select one row. By clicking **Up** or **Down** buttons, you can move it to another position in the list.

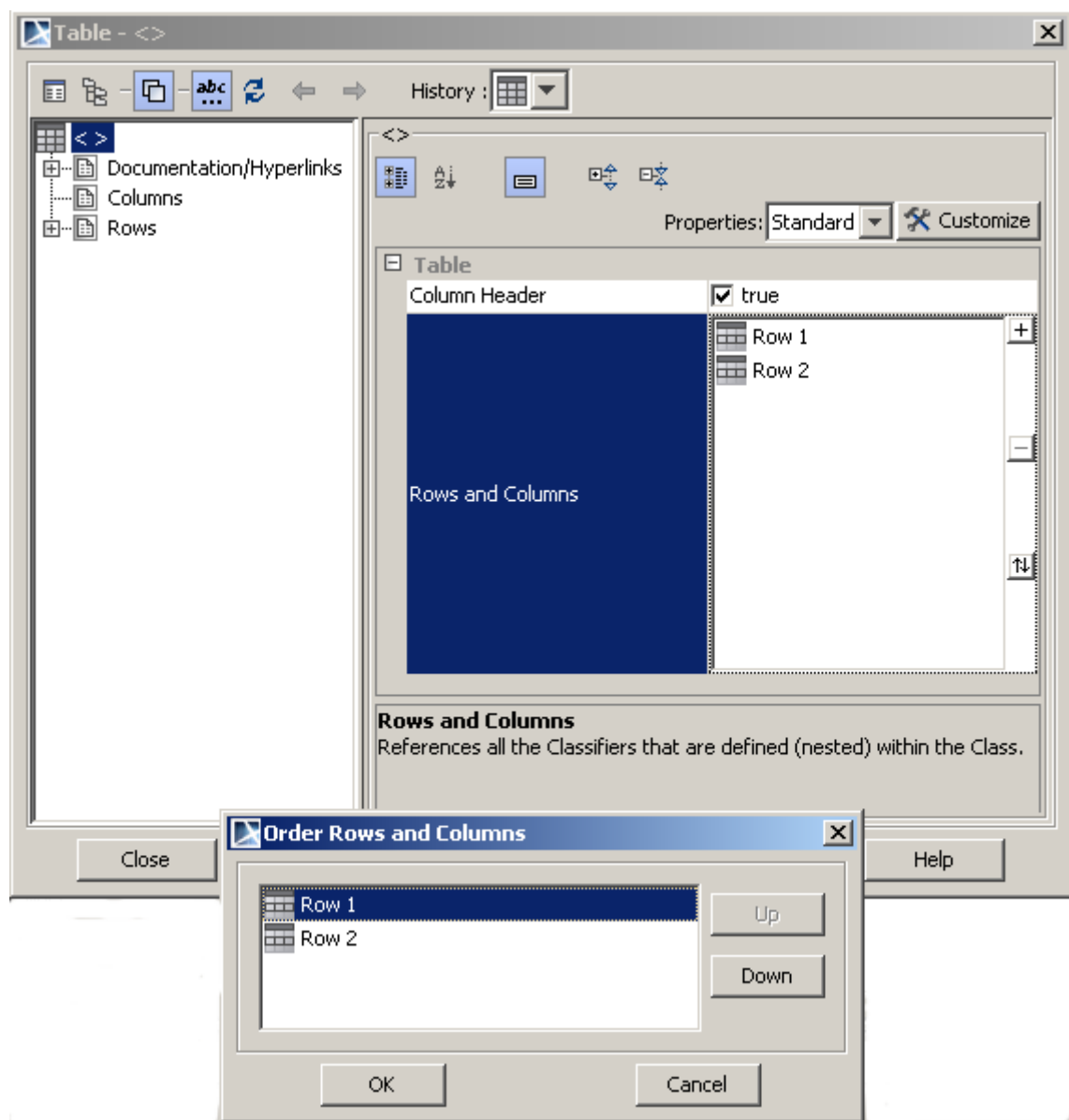


Figure 333 -- Example of How to Order the Rows of a Table

Adding a new column

To add a new column to a table:

1. Select the table in the diagram.
2. Right-click on it and select **Specification** in the shortcut menu.
3. Go to **Columns** node - all columns are listed there.
4. By clicking **Create** button, you can add new column to the table and also one new cell to every row.
5. Type title for the new column.

To change the order of the columns:

1. Select the table in the diagram.
2. Right-click on it and select **Specification** in the shortcut menu.
3. Go to **Columns** node - all columns are listed there.
4. Select one column. By clicking **Up** or **Down** buttons, you can move it to another position in the list.

Adding a value to a table

To adding a value to a table

1. Expand the Tables element in the Browser tree. You will see Cell elements listed under Table elements.
2. Change the name of the Cell:
 - 2.1 Select a Cell element in the Browser and press **F2**. The Cell text area will turn to editable.
 - 2.2 Slowly double click on the Cell in the Browser. The Cell text area will turn to editable.
 - 2.3 Right-click on the Cell element and select **Specification**. The **Cell** specification dialog box will open. Specify the **Text** property and click **Close**.

To add a value to the table, you need to edit the text property of the cell.

Modifying a Tree

You can add as many nodes and leaves as you need to a tree and to any node in the tree and also change the order of the elements of a tree.

To add a node or a leaf to the tree or to another node:

1. Right click on the tree or a node element in the model browser.
2. Select **New Element** and then **Node** or **Leaf**.
3. Type in a text property for the element. This will define the text of the node or the leaf in the diagram.

To change the order of the elements of a tree:

1. Select the table in the diagram.
2. Right-click on it and select **Specification** in the shortcut menu.
3. Go to Nodes and Leafs node - all rows are listed there.
4. Click the small button for Alphabetical View.
5. Select one row. By clicking **Up** or **Down** buttons, you can move it to another position in the list. Click **OK** and the order will be changed.

User Interface Modeling provides a number of predefined buttons which are used quite often. Some of these buttons own one of the icons listed in Table 5, “User Interface Modeling Components,” on page 723 (for use in a toolbar for example) and some own a text (for example **OK** or **Cancel**).

Nesting

As with standard UML components, you can move and arrange user interface components since the components nest in each others and thus create deep structured User Interface models. Not every component, however, can nest in another: only **Container** components can nest other components in them. To know which components can nest, go to Table 5, “User Interface Modeling Components,” on page 723; any component marked with * is a **Container** component.

As Figure 334 on page 734 shows, all frames nest in each others and, as a result, if you were to move the component with title *Frame 1*, all other components would stay in position in this component.

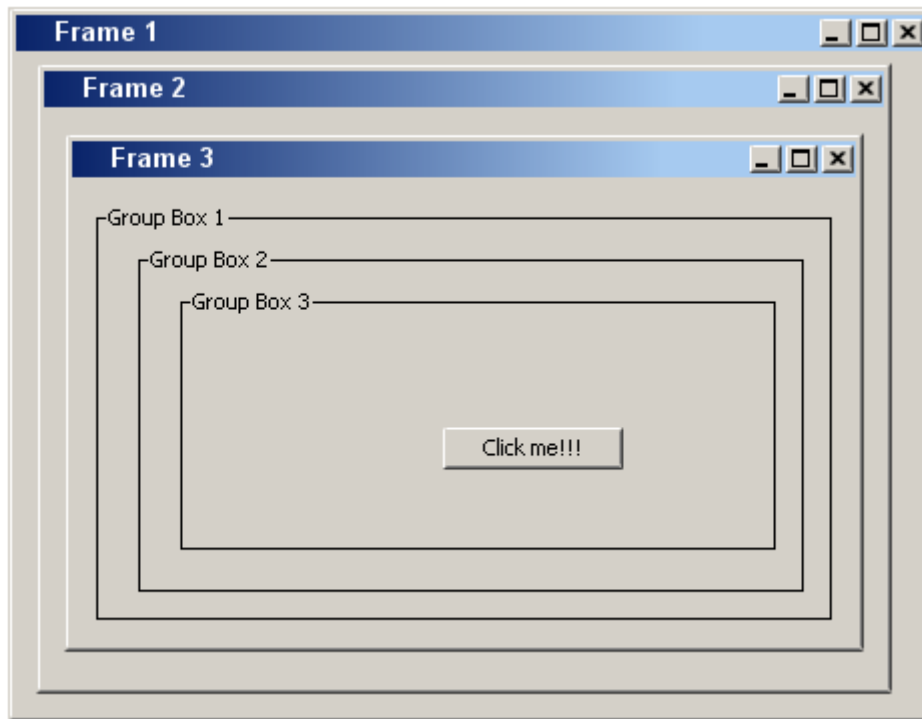


Figure 334 -- Example for a Deep Structure Because of Nesting Components

Reusability

As is the case with standard UML elements, you can also copy and paste user interface components. So, if you have created a complex model and need it again, there is no need to create a new one: simply copy it and reuse it. All you need to do is to select the components; indicate where all the other components are nested on the diagram surface, copy them and then paste them in the same diagram again or in any other diagram. You can also copy just one single component in the same manner or by dragging out of the model browser onto the surface.

Specifying Elements

You can modify User Interface components in different ways. You can edit them through the properties in their specification, edit some properties via the context menu or edit this property on a diagram.

Editing User interface components through the properties in their specification

To edit through the properties:

1. Select a component on the diagram.
2. Right click it and select **Specification** in the shortcut menu (Figure 335 on page 735).
3. Edit one of the properties listed there to modify the component. To see which effect the editing of a property has, see Table 5, “User Interface Modeling Components,” on page 723.

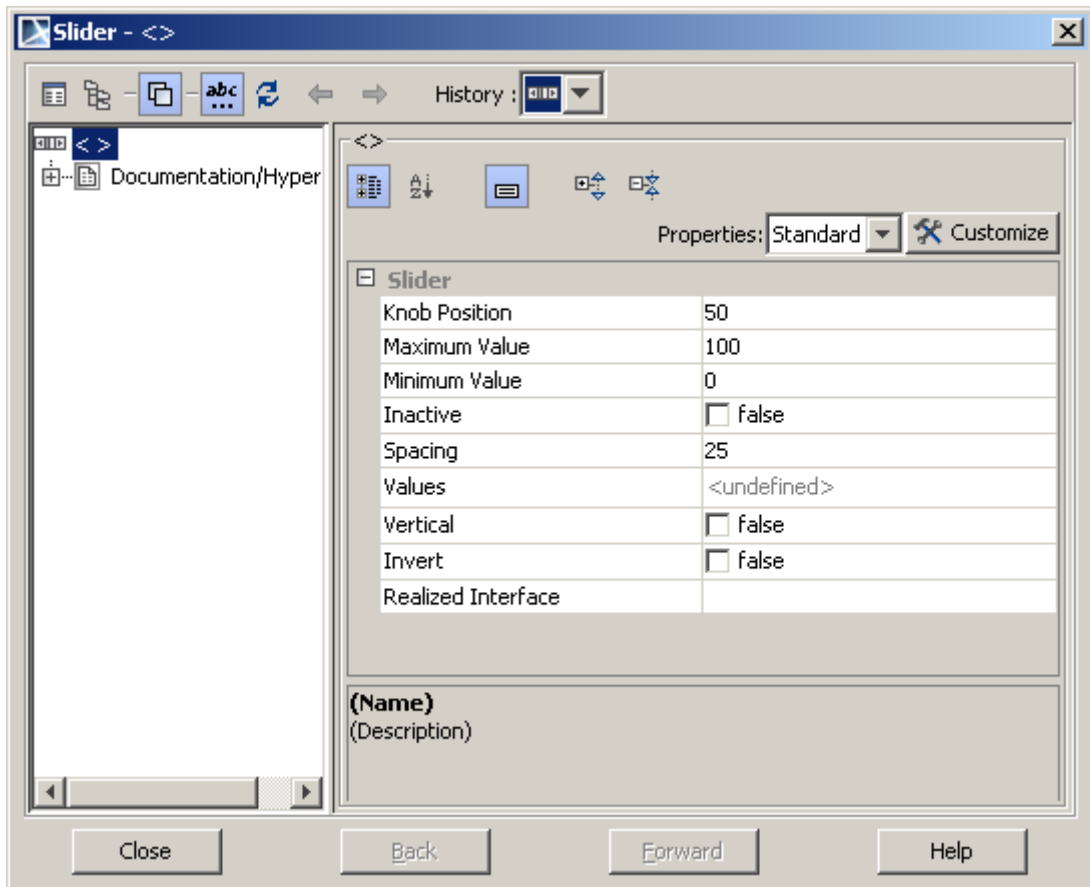


Figure 335 -- Specification Dialog of a Slider component

Editing some properties via the shortcut menu

This is possible for boolean properties (properties where you can only choose between two values for it).

It's also possible to edit some properties via the context menu. This is possible for boolean properties, this means properties where you can just choose between two values for it:

1. Select a component on the diagram.
2. Right click on it. The shortcut menu will appear and the properties which can be edited will be displayed at its bottom (Figure 336 on page 736).
3. Simply click on the property you want to change.

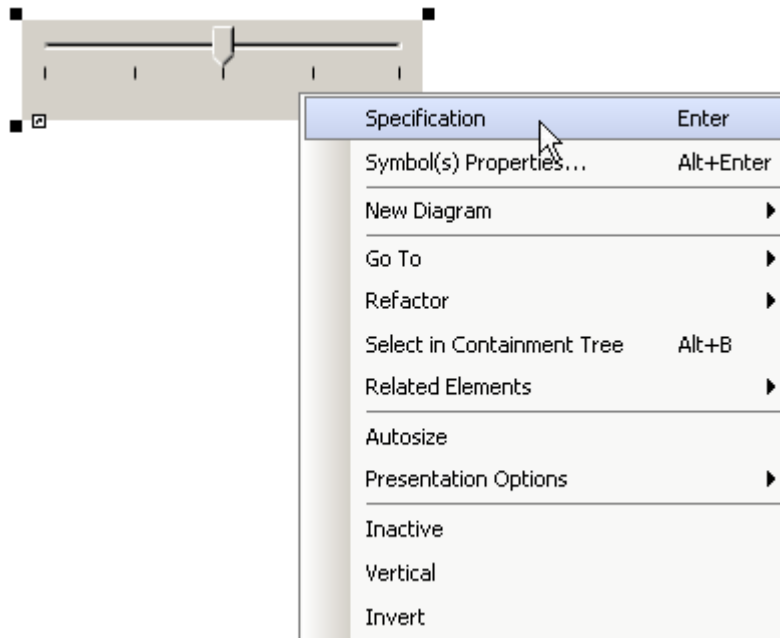


Figure 336 -- Shortcut Menu of the Slider Component from Figure 335 on page 735 with Possible Properties at the Bottom

Editing property on diagrams

For components which own a text or a title, for example a button or a frame, it's also possible to edit this property on a diagram.

To edit on a diagram:

1. Select a component on the diagram.
2. Begin typing in the text which should appear on the component (Figure 337 on page 737).
3. Press **Enter** and the text will appear on the component and also in the property **Text** of the component.

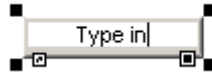


Figure 337 -- On Diagram Editing

Icon Usage

Some components are capable of owning an icon. To that end, User Interface Modeling provides a number of frequently-used icons such as cut, delete or undo/redo, etc. Of course, it is also possible to select any images from your computer as icons.

To add an icon:

1. Select a component which can own an icon, a button for example. To find out which components can have an icon, see Table 5, "User Interface Modeling Components," on page 723.
2. Right click on the component and select **Specification** in the shortcut menu.
3. Select one of the options from property **Icon**. The chosen icon will appear on the component (Figure 338 on page 738). If you choose **custom**, a dialog will appear where you can browse your computer for any image you want to use as an icon.

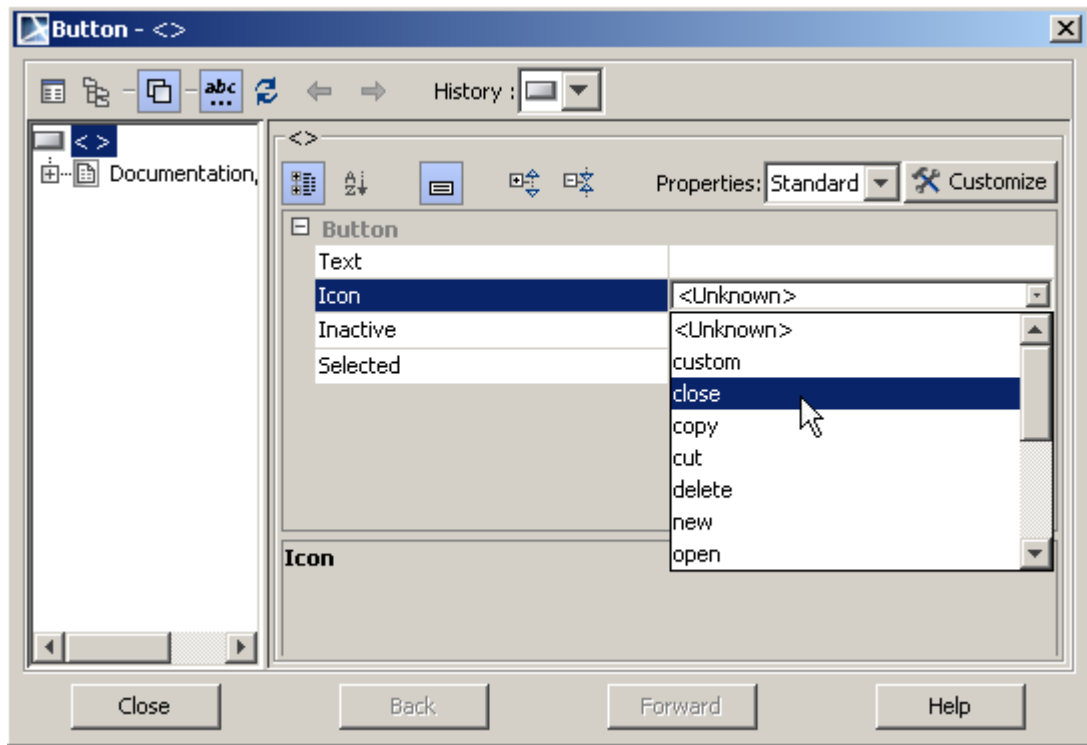


Figure 338 -- Setting an Icon for a Button

Using Symbol Properties

User Interface Modeling also enable you to modify user interface components via symbol properties.

To edit a component via symbol properties:

1. Select a component on the diagram surface.
2. Right click and select **Symbol(s) Properties** in the shortcut menu.
3. Change one of the properties listed in the table below.
4. If you want to set, for example, the same background color to more than one component you can select all these components in the diagram and then change the required property for all of them in the same way.

Table 6 on page 739 lists the symbol properties that are supported:

TABLE 6. Supported Symbol Properties

Symbol Property	Effect
Fill Color	Sets the background color of the component to the chosen one, property Use Fill Color must be marked.
Font	Sets the font if there is any text in the component.
Text Color	Sets the text color of the component to the chosen one.
Pen Color	Sets the color of the border for components that have one.

User Interface Prototyping

User Interface Modeling enables you to create browsable reports. All you need to do to create a browsable report is add a hyperlink to the component of a User Interface model and link it to another User Interface model.

Once a report of these models has been created, click on any component in the report and you will be directed to the other diagram. As Figure 340 on page 742 shows, when you click the **OK** button in the upper left-hand frame you will be directed to the next frame. And using the **OK** button of this frame you can get to a couple of frames in which you can step forward and backward as you want since they are all linked to one another.

“Case Study 3 - User Interface Prototyping Example” on page 749 provides an example of how this feature works and explains how to add hyperlinks and create browsable reports.

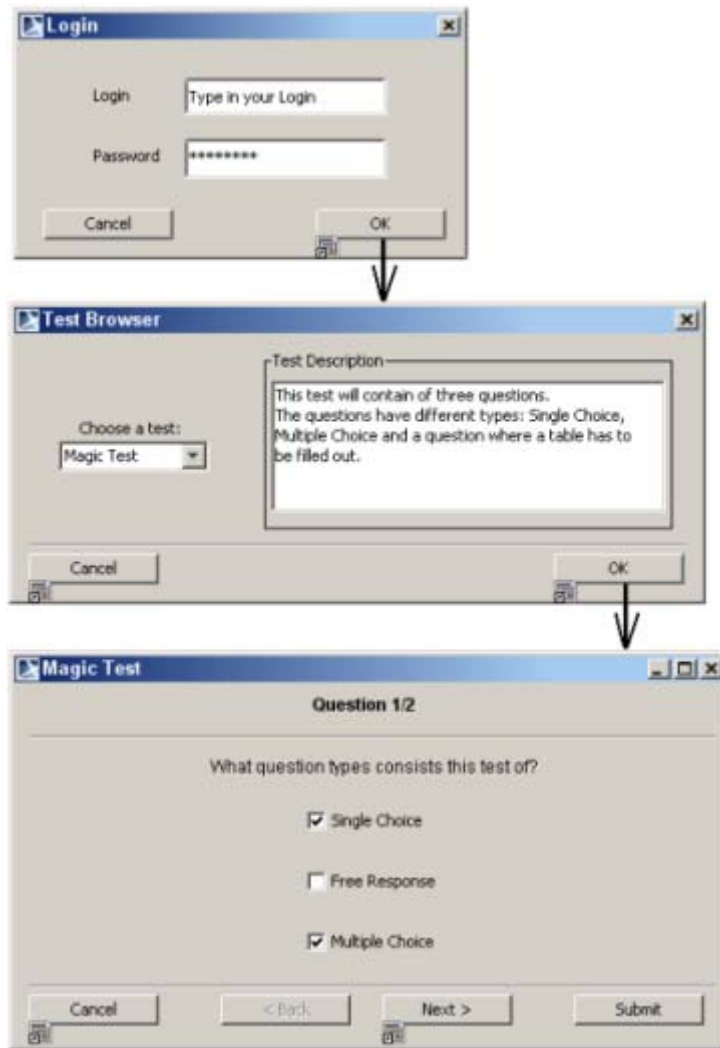


Figure 339 -- Example of User Interface Models and their Workflow

Case Studies for User Interface Modeling

This section includes three case studies on how to create User Interface models:

- “Case Study 1 - Modelling User Interface for the Report Wizard Window” on page 741.
- “Case Study 2 - Slider Example” on page 746.
- “Case Study 3 - User Interface Prototyping Example” on page 749.

Full detailed samples can be found in “<MagicDraw installation folder>\samples\diagrams\User Interface Modeling\UI Modeling Samples.mdzip”.


Case Study 1 - Modelling User Interface for the Report Wizard Window

This case study provides step-by-step instructions for modeling the MagicDraw Report Wizard window.

Step 1# Create a new Project

1. Choose **New Project** from the **File** menu.
2. Name it *Report Wizard*.

Step 2# Create a new Diagram

1. Click the **User Interface Diagram**  button in the main diagram toolbar.
2. Name the new diagram *Report Wizard*.

Step 3# Create Container components

1. Click the **Frame** button in the User Interface diagram toolbar and drag-drop it on the diagram pane. A Frame component will be created.
2. Name the Frame component.
 - Select Frame on the diagram and type its name *Report Wizard*.
 - Or you can define the Frame component name in the **Frame** specification dialog box:
 - 2.1 Double-click the Frame component on diagram to open the Frame specification dialog box.
 - 2.2 Type *Report Wizard* in the **Title** field.
3. Uncheck properties **Maximize** and **Minimize** of the Frame in the **Frame** specification dialog box.
4. Define Icon for the Frame component:
 - 4.1 Select the **custom** option in the **Icon** property in the **Frame** dialog box. The **Open** dialog box will open.

4.2 Browse to the "<MagicDraw installation>\plugins\com.nomagic.magicdraw.uimodeling" folder, and select the *nomagic.png* image to set the frame icon.

5. Create two **Group Boxes** and one **Separator**.

5.1 Name one **Group Box** as *Select Template*

5.2 For the other Group Box uncheck property **Titled**.

5.3 Drag all three components to the **Frame** and arrange everything like in Figure 341 on page 744.

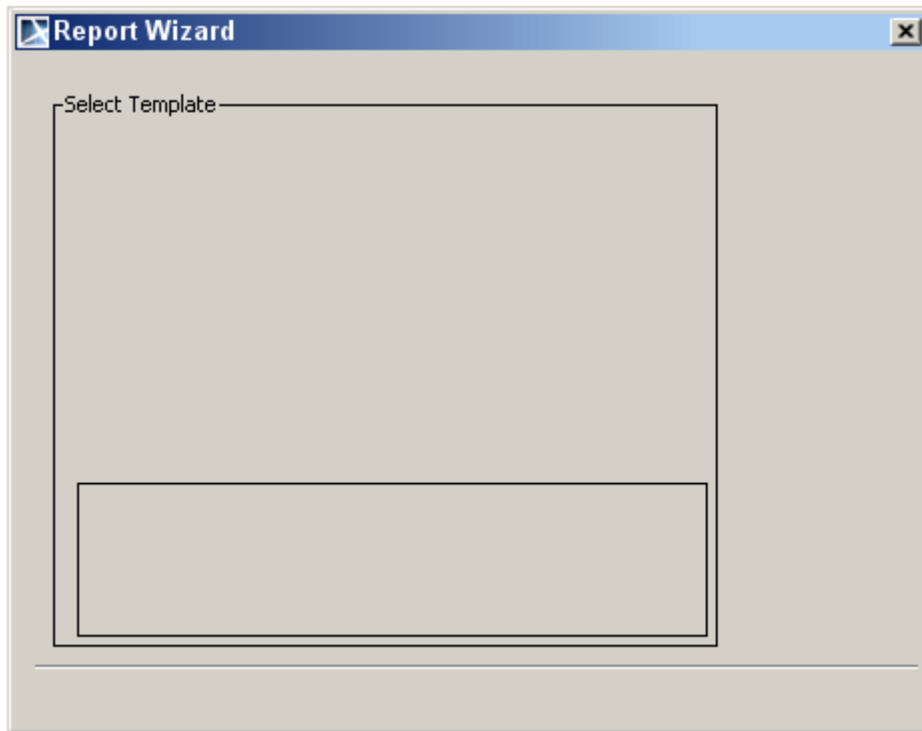


Figure 340 -- Report Wizard Frame with Group Boxes and Separator

Step 4# Create a Tree

1. Create a new Tree component:

1.1 Click the **Other** group in the diagram toolbar to expand the toolbar.

- 1.2 Click the **Tree** button in the diagram toolbar and drag-drop it on the *Select Template* Group Box on diagram pane.
2. Delete old nodes from the Tree component.
3. Create new nodes to the Tree component.
 - 3.1 Right-click on the Tree in the Containment tree to invoke its shortcut menu and choose **New Element > Node**.
 - 3.2 Create five new **Nodes** in this way and name them as shown in Figure 342 on page 745.
4. Add Leaves to Nodes.
 - 4.1 You can add a **Leaf** by right-clicking on a **Node** in the containment tree and then choose **New Element > Leaf**.
 - 4.2 Add at least one **Leaf** to every **Node** to indicate that the nodes have internal elements.

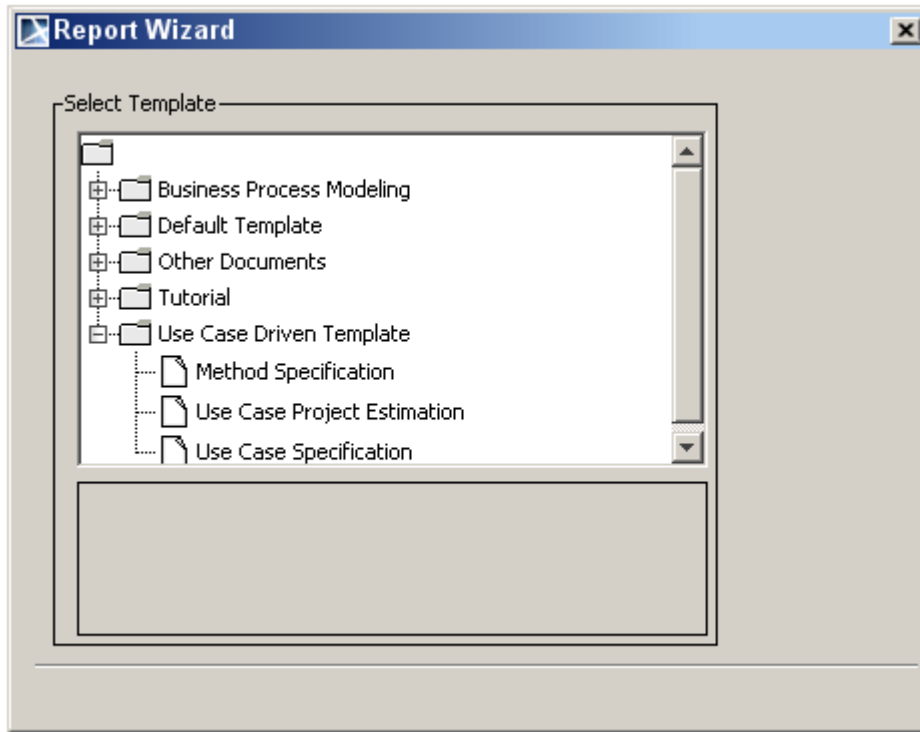


Figure 341 -- The Report Wizard Window with Added Tree

Step 5# Add Buttons

1. Create predefined text buttons.
 - 1.1 Click the **Buttons** group in the diagram toolbar to expand the toolbar.
 - 1.2 Right-click the **Back** button to expand the text buttons group. Create **Back**, **Next** and **Cancel** buttons (Figure 343 on page 747).
2. Create regular buttons.
 - 2.1 Click the **Button** button in the toolbar and drag it to the surface.
 - 2.2 Select the created button in the diagram and type in **New**.
 - 2.3 Create remaining buttons by repeating steps 2.1 and 2.2.
3. Check property **Inactive** for all buttons, except **Next>**, **Cancel**, **New** and **Import**.

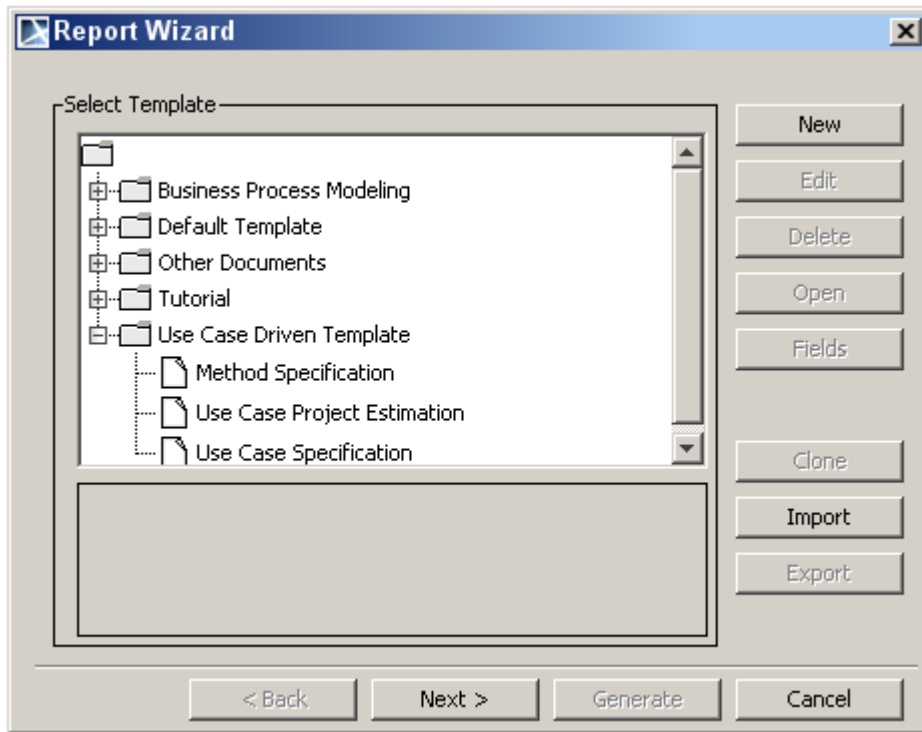


Figure 342 -- Sample of the Report Wizard window

Step 6# Using the Report Wizard window

For example this user interface model could now be exported as an image. The steps to do this are as follows:

1. Select **Save As Image** from the **File** menu.
2. In a new dialog mark **Active Diagram**.
3. In **Image File** define the location where the image should be placed.
4. Select *Joint Photographic Experts Group (*.jpg)* in **Image Format** and then press the **Save-**Button. Of course you can also take another format if you want to.

Case Study 2 - Slider Example

This case study contains step-by-step instructions showing how to create a User Interface model with Sliders. It also shows how to customize the symbol properties of User Interface components. It does not explain, however, how to create a new project and a new diagram since those are explained in “Case Study 1 - Modelling User Interface for the Report Wizard Window” on page 741.

Step 1# Create a Frame, Labels and Sliders

1. Create a new **Frame** and add title *Symbol Properties Customization*.
2. Add new **Label** to the **Frame**.
 - 2.1 Click the **Text** group in the diagram toolbar to expand the toolbar.
 - 2.2 Click the **Label** button and place the label on the diagram. Name it *Fill Color*.
 - 2.3 Create the remaining two labels.
3. Add **Sliders** to the **Frame**.
 - 3.1 Click the **Slider** button in the **Other** toolbar.
 - 3.2 Set property **Spacing** to 50 so as to just have three values marked in the slider.
 - 3.3 Set property **Knob Position** to 0 to move the knob to the left position.
 - 3.4 Select given value *0 0* out of property **Values** and rename it to *0 Red*. Rename the other two given values to *50 Blue* and *100 White*.*
 - 3.5 Draw two more sliders and repeat last steps setting values as shown in Figure 343 on page 747.

* NOTE

With regard to setting the values of a slider it is important to know that there is no empty space between 0. 0, for example, represents a new line. So, in this case, when entering a value, it should look like this:

0

0

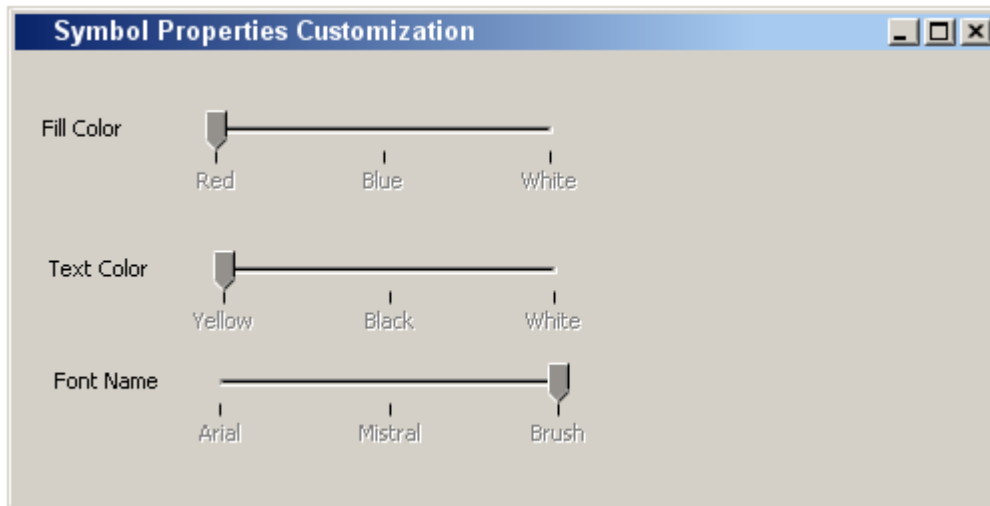


Figure 343 -- Symbol Properties Customization Frame with Added Labels and Sliders

Step 2# Add Text Fields

1. Create four **Text Fields**.

1.1 Click the **Text Field** button in the **Text** toolbar.

1.2 Name it *Red Background*.

1.3 Repeat the previous two steps for the remaining two text fields (Figure 344 on page 748).

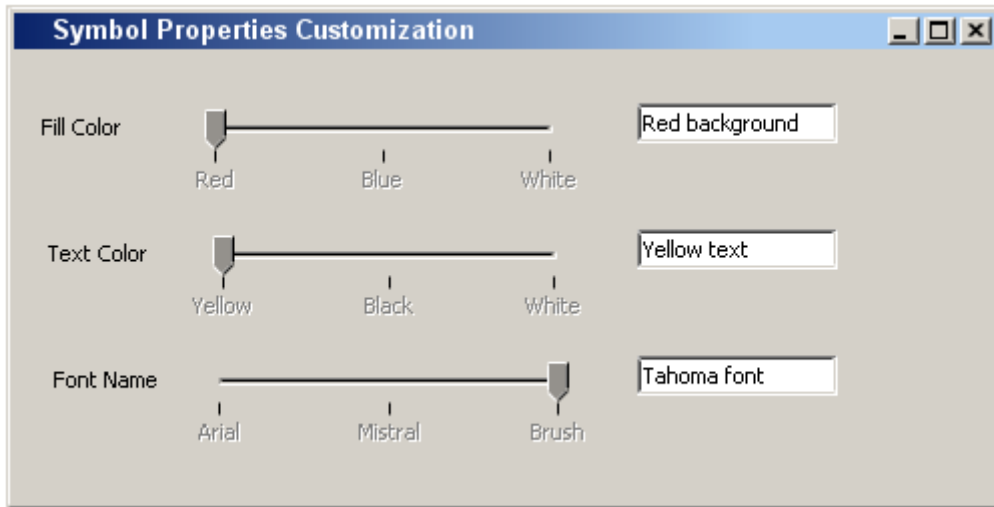


Figure 344 -- Added Text Fields to the Frame

Step 3# Edit Symbol Properties for the Text Fields

1. Edit Symbol Properties for the first text field.
 - 1.1 Select the first text field on the diagram surface.
 - 1.2 Right click and select **Symbol(s) Properties** in the shortcut menu.
 - 1.3 Check property **Use Fill Color** and change **Fill Color** to red.
2. Edit Symbol Properties for the remaining fields.
 - 2.1 For the second text field, select yellow in **Text Color**.
 - 2.2 For the third one, select the font name Tahoma in **Font** (Figure 345 on page 749).

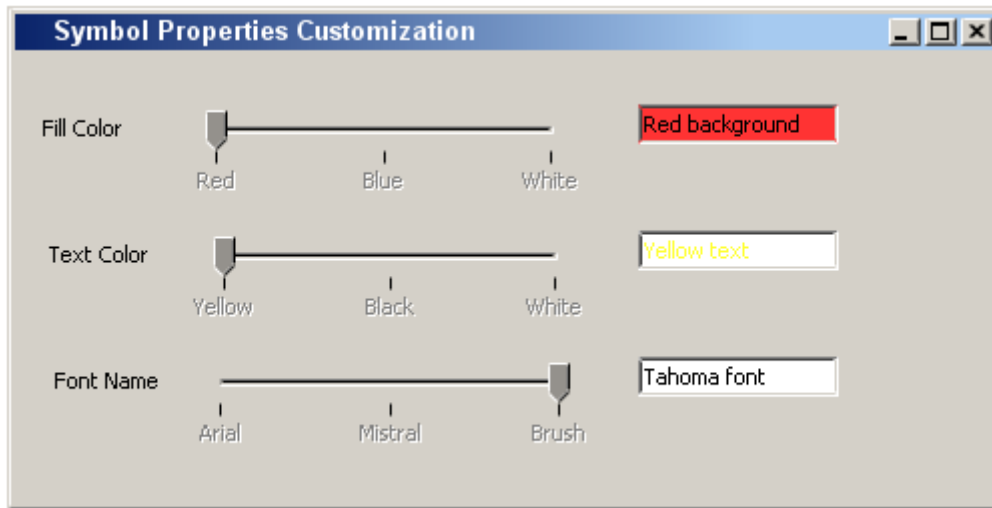


Figure 345 -- Finalized Slider Example

Case Study 3 - User Interface Prototyping Example

This case study shows how to connect several user interface models with one another and create a browsable report out of them to display the wildfowl of an application. The models should represent a test application with a Login Dialog, a Test Browser and a test with several questions. Here are step-by-step instructions for adding hyperlinks and creating browsable reports.

Step 1# Create first Model

1. Create a new **Package** for the model.
2. Create a model similar to the one in Figure 346 on page 750.

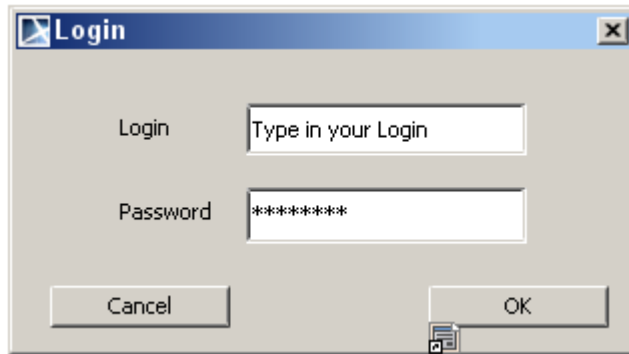


Figure 346 -- First Model of Prototyping Example - Login Dialog

Step 2# Create second Model

1. Create again a new **Package** for this model.
2. Create a model similar to the one in Figure 347 on page 750.

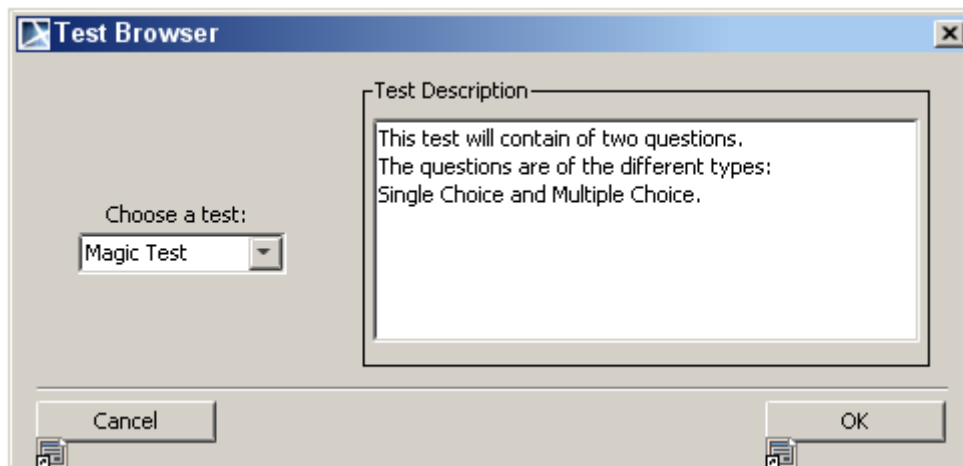


Figure 347 -- Second Model of Prototyping Example - Test Browser

Step 3# Create remaining Models

1. Create a separate **Package** for every model.

2. Build models similar to the ones shown in the full detailed sample which was mentioned in the beginning of this chapter. Or create models for questions on your own.

Step 4# Add Hyperlinks

1. Add a hyperlink.
 - 1.1 Open model with **Login Dialog**.
 - 1.2 Select the **OK** button.
 - 1.3 Click on smart manipulator **Hyperlinks/Go To** and select **Add Hyperlink** in the popup menu.
 - 1.4 Select **Element/Symbol** and click on "..." button. Browse to the **Package Test Browser**, select the **User Interface Diagram** in it and confirm two times with OK - a diagram symbol will appear next to the **OK** button. Double clicking on the **OK** button will lead to the other diagram.
2. Connect now all the buttons in the other diagrams with hyperlinks - you will see which component has an hyperlink because of the diagram symbol next to it.

Step 5# Create a browsable Report

1. Create a new report.
 - 1.1 Select **Report Wizard** from the **Tools** menu.
 - 1.2 Open the node **Default Template** in the tree and select **Web Publisher 2.0**.
 - 1.3 Confirm three times by clicking **Next** until dialog appears where to add data to the report. Add then all **Packages** that contain the created diagrams and click **Next**.
 - 1.4 Give a name to the output file by entering **Report file** and check the box **Display in viewer after generating report**. After clicking **Generate**, report will be built and shown in your default browser.


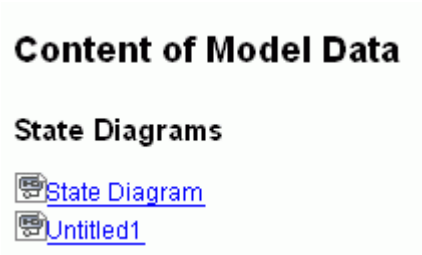






Content Diagram

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.











The Content Diagram is an extension of UML notation. The purpose of the content diagram is to generate or represent a project structure (diagrams) and relations between them. The content table works as a table of contents for a project.

All created diagrams have their own Specifications, in which you can specify their name, documentation, and view the relationships in which they participate. You can also add stereotypes, tagged values and constraints.



Content Diagram Elements

Toolbar Button	Button (Hot key)	Notation
Content Shape Creates a table of contents of all diagrams of the project.	 (C)	
Package	 (P)	
Class Diagram Creates a class diagram.		
Use Case Diagram Creates a use case diagram.		

Toolbar Button	Button (Hot key)	Notation
Communication Diagram Creates a communication diagram.		 Communication Diagram
Sequence Diagram Creates a sequence diagram.		 Sequence Diagram
State Diagram Creates a state diagram.		 State Diagram
Activity Diagram Creates an activity diagram.		 Activity Diagram
Implementation Diagram Creates an implementation diagram.		 Implementation Diagram

Toolbar Button	Button (Hot key)	Notation
Composite Structure Diagram Creates a composite structure diagram.		 Composite Structure Diagram
Time Diagram Creates a time diagram.		 Time Diagram
Struts Diagram Creates a struts diagram.		 Struts Diagram
Web Diagram Creates a web diagram.		 Web Diagram
Business Process Diagram Creates a business process diagram.		 Business Process Diagram

Toolbar Button	Button (Hot key)	Notation
CORBA IDL Diagram Creates a CORBA IDL diagram.		 CORBA IDL Diagram
WSDL Diagram Creates a WSDL diagram.		 WSDL Diagram
XML Schema Diagram Creates an xml schema diagram.		 XML Schema Diagram
Robustness Diagram Creates a robustness diagram.		 Robustness Diagram
Generic DDL Diagram Creates a DDL diagram.		 DDL Diagram
Oracle DDL Diagram Creates an Oracle DDL diagram.		 Oracle DDL Diagram

Toolbar Button	Button (Hot key)	Notation
Content Diagram Creates a content diagram.		

Robustness Diagram

NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

The Robustness Analysis involves analyzing the narrative text of use cases, identifying a first-guess set of objects that will participate in those use cases, and classifying these objects based on the roles they play.

- Boundary or Interface objects are what actors use in communicating with the system.
- Entity objects are usually objects from the domain model.
- Control objects (are usually called controllers because they often are not real objects), serve as the “glue” between boundary objects and entity objects.










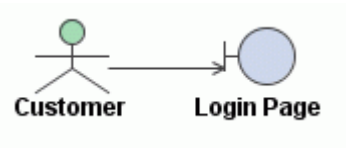
The Robustness analysis serves as a preliminary design within the project life cycle and provides the missing link between an analysis and a detailed design.

Four basic rules apply:

2. Actors can only talk to the boundary objects.
3. The boundary objects can only talk to the controllers and actors.
4. The entity objects can only talk to the controllers.
5. The controllers can talk to the boundary objects and entity objects, and to other controllers, but not to the actors.

Both the boundary objects and entity objects are nouns, and the controllers are verbs. Nouns cannot talk to other nouns, but verbs can talk to either nouns or verbs.

Robustness Diagram Elements

Toolbar Button/Model Element	Button (Hot key)	Notation
Actor An actor represents a role played by an external person, a process or a thing interacting with a system. One physical object may play several roles.	(A) 	
Boundary Actors use the boundary objects to communicate with the system (sometimes called the interface objects.). A class with a stereotype "boundary"	(B) 	
Control Serves as the "glue" between the boundary objects and the entity objects. A class with a stereotype "control"	(C) 	
Entity The entity objects usually are objects from the domain model. A class with a stereotype "entity"	(E) 	
Robustness Association The Association with a default Association End A navigability = false and Association End B navigability = true values.	(S) 	

Web Diagram

NOTE

This functionality is available in Standard, Professional, Architect, and Enterprise editions only.








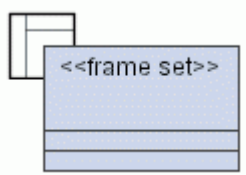
The web system consists of server applications, network, communicating protocol, and the browser. Basically, a user's requests begin from starting the browser and requesting a document through a network from the server (host computer). The web server running on the host computer, catches the user's request, locates the document and delivers it to the user.








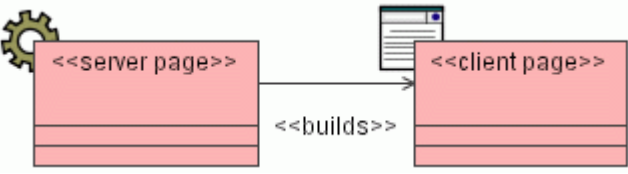

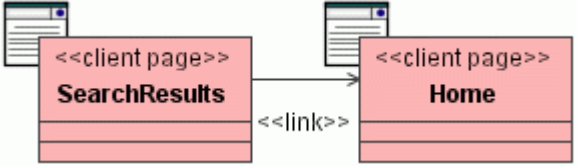
UML is a standard language for modeling software. However, modeling specific web components cannot be done by using just a standard UML. The Web-UML diagram provides an extension to the UML model, which enables developers to model web applications.



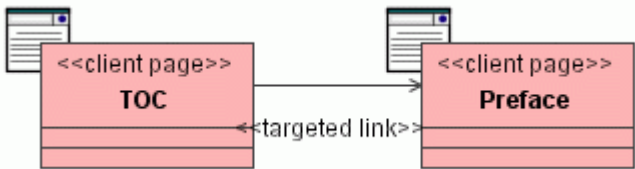

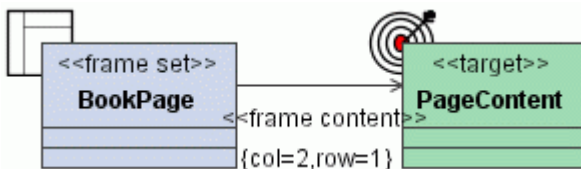

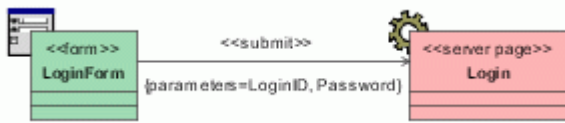


The MagicDraw Web-UML diagram includes Web-UML elements (stereotyped UML elements) for modeling: client, server pages, web form, frame classes, java script class representation and target class, and web page component.


Reference: [Building Web Applications with UML](#) by Jim Conallen Copyright ©2000 by Addison Wesley Longman, Inc.

Web Diagram Elements

Toolbar Button/ Model Element	Button (Hot key)	Some examples
Client Page	 (SHIFT+G)	
Server Page	 (SHIFT+S)	
Form	 (SHIFT+F)	
Frame	 (SHIFT+E)	

Toolbar Button/ Model Element	Button (Hot key)	Some examples
Target	 (SHIFT+T)	
Java Script	 (SHIFT+J)	
Page	 (SHIFT+P)	
Builds	 (B)	
Link	 (L)	

Toolbar Button/ Model Element	Button (Hot key)	Some examples
Redirect	 (T)	
Targeted Link	 (SHIFT+L)	
Frame Content	 (SHIFT+O)	
Submit	 (U)	
Object	 (O)	
RMI	 (SHIFT+R)	

Toolbar Button/ Model Element	Button (Hot key)	Some examples
IIOP	 (SHIFT+I)	

CORBA IDL Diagram








NOTE This functionality is available in Architect and Enterprise editions only.

The CORBA IDL diagram facilitates the creation of CORBA IDL elements. The following patterns are also available for CORBA IDL: Interface, Value Type, Type Definition, Sequence, Array, Fixed, Union, Enumeration, Struct, and Exception.

For more information about CORBA IDL usage in MagicDraw, see the MagicDraw Code & Database Engineering User Guide, Chapter “CORBA IDL Mapping to UML”.

Reference: UML™ Profile for CORBA™ Specification, Version 1.0, April 2002. http://www.omg.org/technology/documents/formal/profile_corba.htm

CORBA IDL Diagram Elements

Toolbar Button/ Model Element	Button (Hot key)
CORBAModule	 (M)
CORBA IDL Interface	 (U)
CORBA IDL Value	 (V)
Generalization	 (G)
Truncatable Generalization	 T
Value Supports Generalization	 Us
CORBA IDL Association	

Toolbar Button/ Model Element	Button (Hot key)
Interface	 (I)



Generic DDL Diagram







NOTE

This functionality is available in Architect and Enterprise editions only.

The DDL diagram is used to draw database definition elements. It simplifies the creation of primary key, foreign key, triggers, etc.

For a detailed description about DDL usage in MagicDraw, see MagicDraw Code & Database Engineering User's Guide, Chapter "DDL Script Engineering".

Generic DDL Diagram Elements

Toolbar Button/ Model Element	Button (Hot key)
Table	 (T)
View	 (V)
Relationship	 (R)
Foreign Key	 (F)
Database	 (D)
Schema	 (S)






Oracle DDL Diagram




NOTE This functionality is available in Architect, and Enterprise edition only.

The DDL diagram is used to draw database definition elements. It simplifies the creation of primary key, foreign key, triggers and etc.

For a detailed description about DDL usage in MagicDraw, see MagicDraw Code & Database Engineering User's Guide, Chapter "DDL Script Engineering".

Oracle DDL Diagram Elements

Toolbar Button/ Model Element	Button (Hot key)
Table	 (T)
View	 (V)
Materialized View	 (M)
Package	 (P)
Type	
Relationship	 (R)
Foreign Key	 (F)








Toolbar Button/ Model Element	Button (Hot key)
Scope	
Database	 (D)
Schema	 (S)




WSDL Diagram

NOTE This functionality is available in Architect and Enterprise editions only.

The WSDL diagram is used to draw WSDL elements. It allows the creation of all elements used in the wsdl file, except schema. The schema elements can be created using the XMLSchema diagram. WSDL plugin provides patterns to create binding elements. The WSDL plugin requires XMLSchema plugin.

WSDL Diagram Elements

Model Element	Button (Hot key)
WSDLmessage	 (M)
WSDLporttype	 (T)
WSDLbinding	 (B)
WSDLport	 (SHIFT+P)
WSDLservice	 (S)
WSDLdefinitions	 (D)
WSDLtypes	 (Y)

Model Element	Button (Hot key)
WSDLimport	 (I)
Xmlns	 (SHIFT+L)
XSDnamespace	 (P)

XML Schema Diagram

NOTE This functionality is available in Architect and Enterprise editions only.








The purpose of the diagram is to create the structure of the xml schema file. It helps to draw the xml schema elements quickly. These elements are the stereotyped UML elements from the class and implementation diagrams.








The following patterns are also available for the XML Schema: XSDcomplexType, XSDcomposition, XSDsimpleType, XSDsimpleType(XSDlist), XSDsimpleType(XSDunion), and Simple XSDrestriction.







For a detailed description about XML Schema usage in MagicDraw, see MagicDraw Code & Database Engineering User's Guide, Chapter "XML Schema".

Reference: <http://www.w3.org/TR/xmlschema-2/>

XML Schema Diagram Elements

Toolbar Button/ Model Element	Button (Hot key)
XSDcomplexType	 (C)
XSDsimpleType	 (S)
XSDsimpleType(XSDlist)	 (SHIFT+S)
XSDsimpleType(XSDunion)	 (U)
XSDall	 (L)
XSDchoice	 (O)
XSDsequence	 (Q)

Toolbar Button/ Model Element	Button (Hot key)
XSDgroup	 (G)
XSDattributeGroup	 (T)
XSDschema	 (SHIFT+M)
XSDnamespace	 (P)
Aggregation	 (A)
Composition	 (F)
Template Binding	 (B)

Toolbar Button/ Model Element	Button (Hot key)
XSDrestriction	 (R)
XSDextension	 (E)
XSDimport	 (M)
XSDinclude	 (D)
XSDredefine	 (SHIFT+F)
xmlns	 (SHIFT+L)






Time Diagram

NOTE

This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

A Time Diagram is an extension of UML notation. The time diagram is similar to a sequence diagram, but the model elements of the time diagram have the predefined stereotypes.

Time Diagram Elements

Toolbar Button/ Model Element	Button (Hot key)
Lifeline <<CRconcurrent>>	
Lifeline <<SAschedRes>>	
Message <<RTevent>>, <<CRimmediate>>	
Message <<CRimmediate>>	
Message <<SAttrigger>>	
)	








Struts Diagram








NOTE



This functionality is available in Professiona, Architect. and Enterprise editions only.

A Struts Diagram is an extension of UML notation.

Struts Diagram Elements

Toolbar Button/ Model Element	Button (Hot key)
Class	 (C)
Class by Pattern	 (SHIFT+P)
Interface	 (I)
ActionForm class for struts	 (G)
Action class for struts	
JavaServer Page for use with struts	
Package	 (P)

Toolbar Button/ Model Element	Button (Hot key)
Model	 (M)
Interface Realization	 (R)
Realization	 (E)
Abstraction	 (T)
Usage	 (U)
Generalization	 (G)
Association	 (S)

Toolbar Button/ Model Element	Button (Hot key)
Aggregation	 (A)
Composition	 (F)











Networking Diagram












NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.


This diagram allows a visual display of the network topology. The *Networking Profile* contains stereotypes for the network description. Elements with icons can be drawn on the Diagram pane.

The Networking Diagram is commonly used to depict hardware nodes as well as the connections between them.

Networking Diagram Elements

Toolbar Button/ Model Element	Button (Hot key)
Server	
Application Server	
DB Server	
File Server	
Proxy Server	
Web Server	
PC	
Laptop	
Monitor	
Fax	

Toolbar Button/ Model Element	Button (Hot key)
Plotter	
Printer	
Scanner	
Modem	
Router	
Switch	
Firewall	
Database	
Program	
Internet Browser	
Document	

Toolbar Button/ Model Element	Button (Hot key)
File	
Wireless Network	
Internet	
Building	
City	
Actor	
User	
HTTP	
Ethernet	
RMI	
Communication Path	

Business Process Diagram

NOTE

This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

BPMN (Business Process Modeling Notation) is the most recognized standard used by business users for an end-to-end business process modeling. MagicDraw with its business-oriented notation support has become one useful tool for both business and IT users. Working with business and IT models in one tool helps save time and solve model interchange problems.








The Business Process Diagrams convey the business processes of different participants in a modeled system. In a MagicDraw model, Process contains the Business Process Diagram, depicting it as a graph of the Flow Objects.









The BP diagrams can be used for three basic BPMN sub-models creation:

- Private (internal) business processes
- Abstract (public) processes
- Collaboration (global) processes


Within and between these three BPMN sub-models, many types of Diagrams can be created (High-level private process activities, detailed private business processes, Collaboration Processes, etc.). Since a BPMN Diagram may depict the Processes of different Participants, each Participant may view the diagram differently.




Business Process Diagram Elements








Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Start Event None		The Start Event indicates where a particular process will start.
Start Event Message		
Start Event Timer		
Start Event Rule		
Start Event Link		
Start Event Multiple		







Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Intermediate Event None		The Intermediate Events occur between a Start Event and an End Event. It will affect the flow of the process, but will not start or (directly) terminate the process.
Intermediate Event Message		
Intermediate Event Timer		
Intermediate Event Error		
Intermediate Event Cancel		
Intermediate Event Compensation		















Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
End Event None		The End Event indicates where a process will end.
End Event Message		
End Event Error		
End Event Cancel		
End Event Compensation		








Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
End Event Link		
End Event Terminate		
End Event Multiple		
Task		The Task is used when the work in the Process is not broken down to a finer level of Process Model detail.
Loop Task		


Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Multiple-Instance Task		
Compensation Task		
Collapsed Sub-Process		A Sub-Process is a compound activity that is included within a Process. It can be broken down into a finer level of detail (a Process) through a set of subactivities.
Collapsed Transaction		
Collapsed Loop Sub- Process		




Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Collapsed Multiple-Instance Sub-Process		
Collapsed Compensation Sub-Process		
Collapsed Ad-Hoc Sub- Process		
Expanded Sub-Process		

Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Expanded Transaction		
Expanded Loop Sub- Process		
Expanded Multiple-Instance Sub-Process		
Expanded Compensation Sub-Process		
Expanded Ad-Hoc Sub- Process		

Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Gateway		A Gateway is used to control the divergence and convergence of multiple Sequence Flows.
Data-Based XOR		
Event-Based XOR		
OR		
Complex		
AND		

Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Sequence Flow		A Sequence Flow is used to show the order of how activities will be performed in a Process.
Conditional Sequence Flow		
Default Sequence Flow		
Compensation Association		The Compensation Association occurs outside the Normal Flow of the Process and is based on an event (a Cancel Intermediate Event) that is triggered through the failure of a Transaction or a Compensate Event.
Message Flow		A Message Flow is used to show the flow of messages between two entities that are prepared to send and receive them.

Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Horizontal/Vertical Pool		A Pool represents a Participant in a Process.
Pool with Lanes		
Data Object		The Data Objects provide information about what activities are required to be performed and/or what they produce.
Association		An Association is used to associate information with the Flow Objects.
Directional Association		




Toolbar Button/ Model Element	Button	Element Description
Text Annotation		
Association of Text Annotation		
Group		A grouping of activities that does not affect the Sequence Flow.



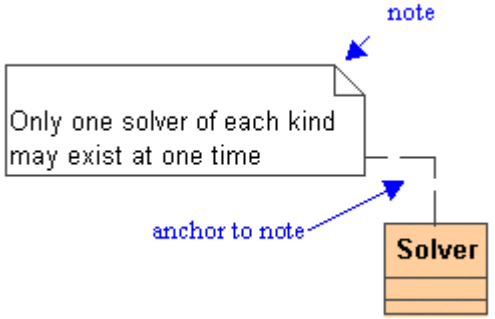



The Business Process diagram possesses smart layout features. For more information about these features, see “Smart Activity Diagram layout” on page 696.


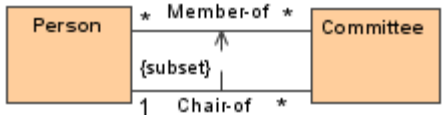

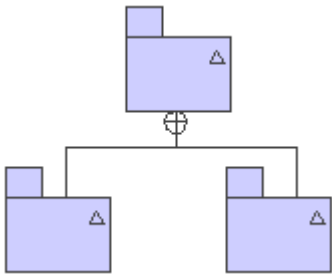



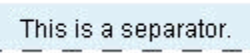
11 MODEL ELEMENTS


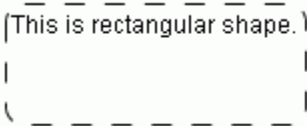
Common Model Elements in the Diagrams

The following symbols are only graphical symbols. They do not possess any data.

Model elements	Button	Function	Notation
Text Box		Type texts directly on the diagram.	
Text Box (HTML text)	(X)		
	 (SHIFT+X)		

Model elements	Button	Function	Notation
Note Note (HTML text)	 (N)  (SHIFT+N)	<p>A graphical symbol containing a textual information. Use a note to add any information needed for your diagram. Usually the note is connected to another symbol using an anchor line.</p>	
Comment Comment (HTML text)	 	<p>A graphical symbol, which gives an ability to display different remarks on the diagrams.</p>	
Anchor	 (H)	<p>Relates symbols and notes. Use an anchor to relate any symbol to a note or comment. The style of the anchor can be changed to rectilinear, oblique, or bezier.</p> <p>For a detailed description on editing the appearance of paths, see "Drawing Relationship Paths" on page 263.</p>	

Model elements	Button	Function	Notation
Constraint NOTE To select a constraint, right-click the Note Anchor button.	(SHIFT+H) 	The presentation of a constraint between two graphical symbols.	
Containment	SHIFT+C 	Shows a class, package or other model elements declared within another model element.	
Dependency	 (SHIFT+D)	Indicates a semantic relationship between two or more model elements.	
Image Shape	(I)	Provides a simple and quick way to insert pictures into a diagram. They can be logo, graph, table, etc.	
Separator	 (W)	Separates different parts of a diagram with a horizontal line.	

Model elements	Button	Function	Notation
Rectangular Shape		Separates different parts of a diagram with a rectangular shape.	

Note, Comment

A note is a graphical symbol containing a textual information. It is used to add any information needed for your diagram. A note is usually connected to another element symbol using an anchor line.

A comment is a graphical symbol used to display different remarks on the diagrams.

To change the text display mode

1. Connect a note to an element symbol with an anchorlink.
2. From the note shortcut menu, select **Text Display Mode** and then **Show Text**, to display the text that is added to the note. Select **Show Documentation**, to display a documentation of the element on the note.

To show the constraints and/or tagged values on the note

1. Connect a note to an element, the constraint and/or tagged value of which is specified, by an anchor link.
2. From the note shortcut menu, select **Show Constraint** and/or **Show Tagged Value**. Separate compartments with the constraint and tagged values will be displayed on the note.

To remove a line between the compartments on the note

1. From the note shortcut menu, select **Symbol(s) Properties**.
2. Clear the **Show Line Between Compartments** check box in the open dialog.

Anchor

Use anchors to relate symbol to a note or comment. You can change the anchor style to rectilinear, oblique, or bezier.

For a detailed description on editing the appearance of paths, see “Drawing Relationship Paths” on page 263.

Constraint path

The constraint for two graphical symbols (such as two classes or two associations) is shown as a dashed arrow from one element to the other. The constraint is labeled by the constraint string (in braces {}). The direction of the arrow represents a relevant information within the constraint. The client (the tail of the arrow) is mapped to the first position and the supplier (arrowhead) is mapped to the second position in the constraint.

To add a constraint expression to the constraint path

From the constraint path shortcut menu, choose **Select Constraint** and select one of the following:

- Select **<new>**. The **Select Extension Element Storage** dialog box opens. Select the folder where the constraint will be stored and click **OK**. The **Constraint Specification** dialog box opens. Fill in the dialog box fields.
- Select an already created constraint from the list.

Image Shape

The Image Shape provides a simple and quick way to insert a picture into a diagram. This can be logo, graph, table, etc. The preferred shape size after the insertion is the actual image size.

To insert a picture on the Diagram pane

1. Click the **Image Shape** button on the Common toolbar. The **Open** dialog box opens.
2. Select an image in *.gif, *.jpg, *.jpeg, *.svg, or *.png format and click **Open**.

Separator

You may use a horizontal separator to rule off different parts of a diagram.

To set the text position of a separator name

- From the separator shortcut menu, select **Text Position**, and then select the text position you need: **Left** (default), **Center**, or **Right**.
- Set the text position in the **Project Options** dialog box.

To set a separator line style (dashed or solid)

- From the separator shortcut menu, select **Line Style**, and then select the style you need: **Dashed** (default) or **Solid**.

Documentation

Define a documentation for various documents in the **Comment Specification** dialog box. You can also define a comment stereotypes, tagged values, and constraints.

To open the **Comment Specification** dialog box

1. In the element **Specification** dialog box, click **Documentation/Hyperlinks**.
2. Expand the **Documentation/Hyperlinks** branch in the **Specification** tree and double click **Comment**.

To display the documentation on the Diagram pane

1. Draw a note on the Diagram pane.
2. Using an Anchor to Note, connect the note to the model element containing the documentation you want to display.
3. From the note shortcut menu, select **Text Display Node** and then **Show Documentation**.

TIP!

If a comment contains stereotypes, tagged values, and/or constraints, you can choose to display them in the content of the note.

UML Extension Elements

UML is a general purpose visual modeling language for specifying, constructing and documenting the artifacts of systems that can be used with all major application domains and implementation platforms. It has been widely adopted by both industry and academia as the standard language for describing software systems. However, the fact that UML is a general purpose notation may limit its suitability for modeling some particular specific domains (e.g. web applications or business processes), for which specialized languages and tools may be more appropriate.

UML provides a set of extension mechanisms (stereotypes, tag definitions, and constraints) for specializing its elements, allowing customized extensions of UML for particular application domains. These customizations are sets of UML extensions grouped into UML profiles.

There are several reasons why you may want to extend a metamodel:

- Assign a terminology that is adapted to a particular platform or domain (such as capturing EJB terminology like home interfaces, enterprise java beans, and archives).
- Assign a syntax for constructs that do not have a notation (such as in the case of actions).
- Assign a different notation for the existing symbols (such as being able to use a picture of a computer instead of the ordinary node symbol to represent a computer in a network).
- Add semantics that is left unspecified in the metamodel (such as how to deal with priority when receiving signals in a statemachine).
- Add semantics that does not exist in the metamodel (such as defining a timer, clock, or continuous time).
- Add constraints that restrict the way you may use the metamodel and its constructs (such as disallowing actions from being able to execute in parallel within a single transition).
- Add information that can be used when transforming a model to another model or code (such as defining mapping rules between a model and Java code).

I MagicDraw goes beyond the traditional profiling. It allows developers to create a new modeling tool based on their UML profile. In the following sections, you will find information about how to work with profiles, stereotypes, and tags.

For a detailed description of UML Profiling and DSL in MagicDraw, see the [UML Profing and DSL User's Guide](#).

| Stereotype

A stereotype defines how an existing metaclass may be extended. It enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass.

Just like a class, a stereotype may have properties, which may be referred to as tag definitions. When a stereotype is applied to a model element, the values of the properties may be referred to as tagged values.

Any model element from the reference metamodel (any UML model element) can be extended by a stereotype. For example in UML, States, Transitions, Activities, Use cases, Components, Attributes, Dependencies, etc. can all be extended with the stereotypes.

The stereotypes are created as separate model elements and can be drawn in the Class diagrams. According to UML, the stereotypes have to be stored in the Profiles (a subtype of Package), so the Class diagram for the stereotypes should be created in the Profile instead of the Package.

For more information about working with symbols, see Chapter “Working With Diagrams” on page 245.

To create a new stereotype

To create a new stereotype it is recommended to first create a profile (see the Section above).

- Use the **Stereotype** button on the class diagram toolbar, **Profiling Mechanism** group. The **Select Metaclass** dialog box opens. Specify the metaclass that you want to extend and click **OK**. Double-click to open the **Stereotype Specification** dialog box. For a detailed description of this dialog box, see Section “Stereotype Specification dialog box” on page 806.
- From the Profile or Package shortcut menu in the Browser, select **New**, and then select **Stereotype**.

To create a stereotype with an image

1. Open the **Stereotype Specification** dialog box.

2. Click the **Icon** “...” button and from the **Open** dialog box, select an image you want to place for the stereotype. Click the **Open** button.

NOTE

For more information about displaying stereotype icon on shape, see “Displaying icon or image on the symbol or instead the symbol” on page 344.

To view assigned image properties

1. In the **Stereotype** specification dialog box, invoke the **Icon** text box shortcut menu.
2. Select the **Open Specification** dialog box. The **Image** specification dialog box opens. You can see the format and location of the image, and specify the other image properties.

To display a stereotype icon as a main shape on the diagram pane

From the shape that has an assigned stereotype with an icon, select **Presentation Options** and then **Shape Image** from the **Shape Shortcut menu**. Note that all compartments must have been suppressed before.

To order stereotypes

Stereotypes can be ordered. Symbol style of the first in the list stereotype will be applied to the symbol on diagram and in Browser.

To order stereotypes:

1. Invoke symbol shortcut menu, select the **Stereotype** command. The list of stereotypes opens (see Figure 348 on page 805).
2. Click the **Order** button. The **Order Stereotypes** dialog box opens (see Figure 349 on page 806).
3. Click **Up** or **Down** buttons to order stereotypes.

11

MODEL ELEMENTS

UML Extension Elements

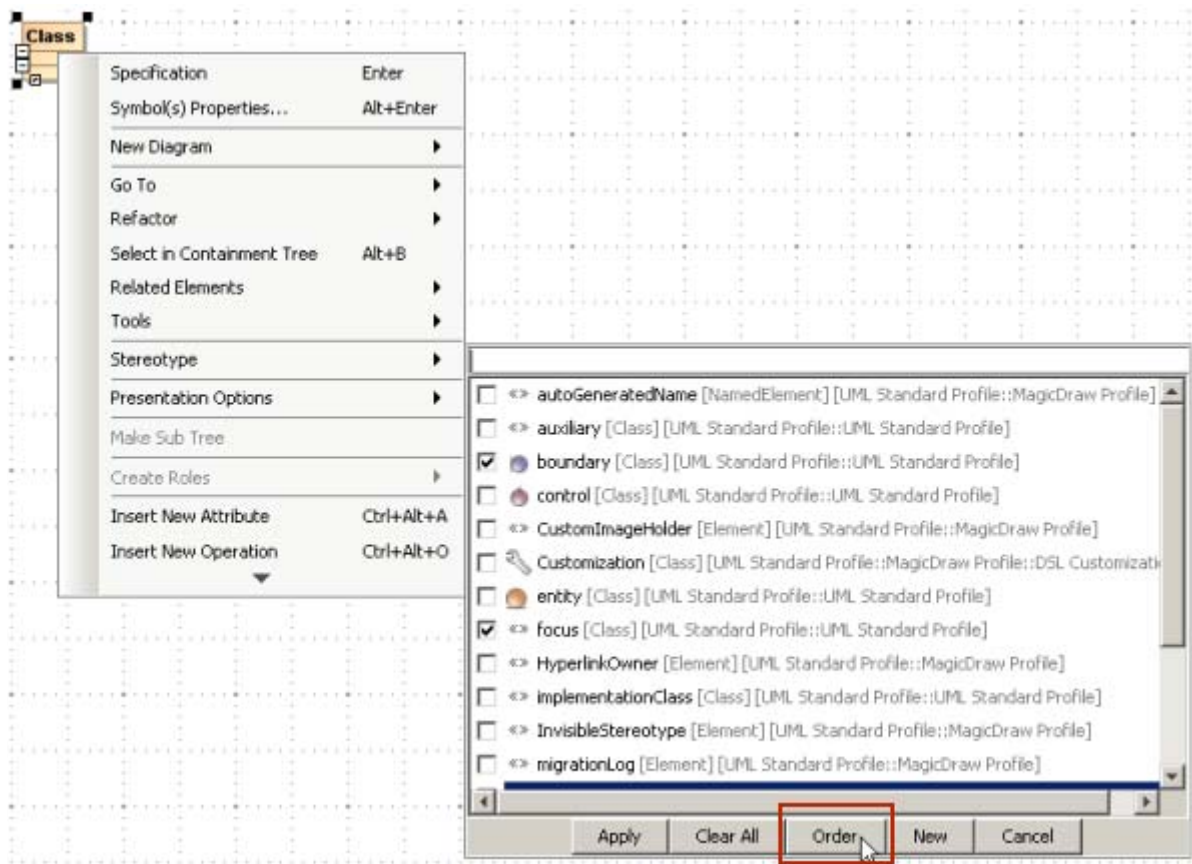


Figure 348 -- The Order button in the Stereotypes list, in the Class symbol shortcut menu

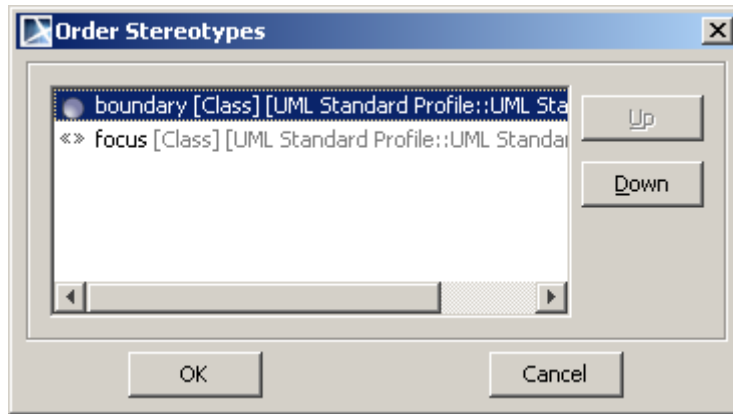


Figure 349 -- The Order Stereotypes dialog box

Saving of stereotype information in XMI

Ability to choose options where to save applied stereotype information in XMI file. Information can be saved at the end of the file or inside the element information.

By default stereotype information is stored at the end of XMI file. To store information inside element, open **Environment Options** dialog (choose **Options - Environment** command from main menu), and in **General - Save/Load** section select property **Save** stereotype information within element.

Stereotype Specification dialog box

Define the selected stereotype in the **Stereotype Specification** dialog box.

To open the **Stereotype** specification dialog box

- Double-click the stereotype shape or select **Specification** from the stereotype shortcut menu.

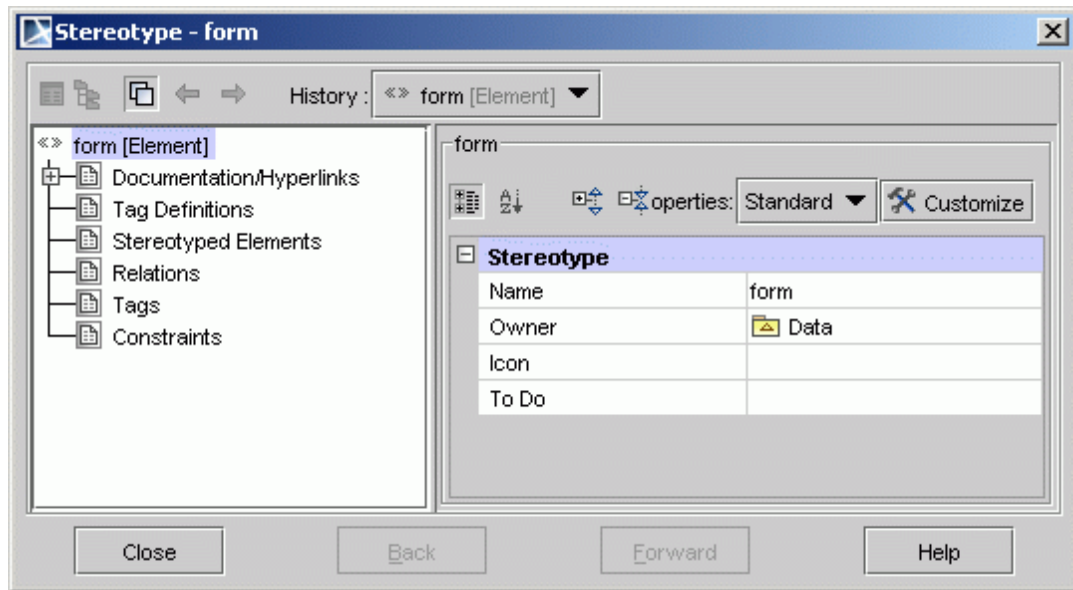


Figure 350 -- Stereotype Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Group name	Box name	Function
General Set a general information about the stereotype.	Icon	An image file icon. Refer to the UML Profiling and DSL manual for the description of the advanced icons for relationships.
Tag Definitions Specify information for tag definition	Name	The name of a stereotype attribute.

	Type	In general case, where the tag type is a data type, this specifies the range of values of the tagged values associated with the tag definition. In a special case, where the tag type refers to a meta-class that is not a data type, the tag value references model elements that are instances of the metaclass.
	Classifier	The classifier element of a stereotype attribute.
	Default value	The default value for tag definition.
	Create	Creates a new tag definition.
	Delete	Deletes an existing tag definition.
	Clone	
	Up/Down	
Stereotyped Elements Shows the model elements affected by the stereotype.	Name	The name of the stereotyped model element.
	Type	The type (class, use case, etc.) of a model element.
	Add	Applies a stereotype to the selected model element.
	Remove	Unapplies a stereotype from the selected model element.

Assigning a stereotype

You may assign a stereotype to an element in the following ways:

- Open the corresponding element **Specification** dialog box. Click the "..." button in the general pane, next to the **Applied Stereotype** property. Select one or more available stereotypes from the open list and click **Apply**. Or you may create a new stereotype by clicking **New**.
- You may apply a stereotype to a model element easily from the model element shortcut menu. To do this, follow these steps:
 1. Select the model element on the diagram pane or in the Browser. Open its shortcut menu by right-clicking the mouse.

2. From the model elements shortcut menu, select the **Stereotype** command. The stereotype menu opens.
3. The list of all stereotypes is seen. You can type the name of the stereotype for which you are searching in the text box, above the list. Or, you can create a new stereotype by clicking the **New** button.
4. When you find the stereotype that you want, or create a new one, select the check boxes of those stereotypes to add to the model element.
5. Click the **Apply** button.
 - On the diagram pane in the element name area, type two open angle brackets <<, type the stereotype name and type two close angle brackets. Then you may type the element name itself. For example: if you want to name element Books and assign <<table>> stereotype, in the element name area type the following: <<table>>Books. The name completion for the stereotypes works in the name editing mode, press the **Ctrl+Space** or **Ctrl+Backspace** to get a list of possible to apply stereotypes.

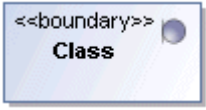
Changing the stereotype display mode

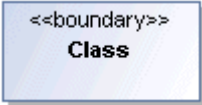




MagicDraw version 15.0 and later has improved the stereotype display functionality. Now you can change the stereotype name and its icon visibility on the element shape instead of on the display stereotype image.

To change the stereotype visibility mode on the element shape:

- From the element shape shortcut menu, select the **Presentation Options > Show Stereotypes** command and then select the desired property mode.
- You may change the stereotype visibility mode in the symbol **Properties** dialog box > **Show Stereotypes** combo box.

Select one of the six property modes for **Show Stereotypes**. The property modes are described in the table below.

Show Stereotypes Property Mode	Shape	Icon of the stereotype on the shape	Name of the stereotype	Image of the stereotype instead of the element shape
Text and icon		displayed	displayed	-

Show Stereotypes Property Mode	Shape	Icon of the stereotype on the shape	Name of the stereotype	Image of the stereotype instead of the element shape
Text		not displayed	displayed	-
Icon		displayed	not displayed	-
Shape Image and Text		-	displayed	displayed*
Shape Image		-	not displayed	displayed*
Do Not Display		not displayed	not displayed	-

* - To display the image of a stereotype instead of the element shape all element compartments should be suppressed.

NOTE **Shape Image and Text** and **Shape Image** properties are not added to the Relationships, Roles and Diagram Frame element properties list.

Parent topic: “Stereotype” on page 803.

Related topics:

“Assigning a stereotype” on page 808.



Stereotype notation

Since v 16.6 MagicDraw supports standard stereotype notations. Stereotype notations in diagrams use the guillemets « » instead of symbols << >> (Figure 351 on page 811).

However, when editing elements in a diagram, you can still enter the stereotype names between the << >> symbols.



Figure 351 -- Stereotype Notation

Tags

Just like a class, a stereotype may have properties, which may be referred to as tag definitions. When a stereotype is applied to a model element, the values of the properties may be referred to as tagged values.

The tag definitions are used to define new meta attributes of the extended metaclass, they are used as regular class attributes. For more information about working with class attributes, see Section "Drag and drop the selected Opaque Behavior element from the Browser tree on the Diagram pane." on page 947.

An actual instance of the tag definition is a tagged value (tag).

A tag holds extra information like:

- additional information that does not come with UML, for example Precondition for Use Cases.
- management data about the state and progress of the project such as author, status, and tested.
- language specific data for code generation tools.

A tagged value consists of two parts: name and value (example: Author = Joe).

To create a new tag definition

1. Create a new stereotype. Open the **Stereotype Specification** dialog box.
2. Click the **Tag Definitions** tab and use the **Create** button to add a new tag definition for stereotype. Select the type of this property. It may be a standard UML data type or another user defined Stereotype. Regular classes should not be used as types of tag definition.
3. Click **Close** in the **Stereotype Specification** dialog box to save changes.

Refer to Section “Stereotype Specification dialog box” on page 806 for more information about **Stereotype Specification** dialog box fields not covered in this section.

Editing Tagged Value

To create a new tagged value

- In the element **Specification** dialog box, **Tags** group, select an available tag definition and click the **Create Value** button.

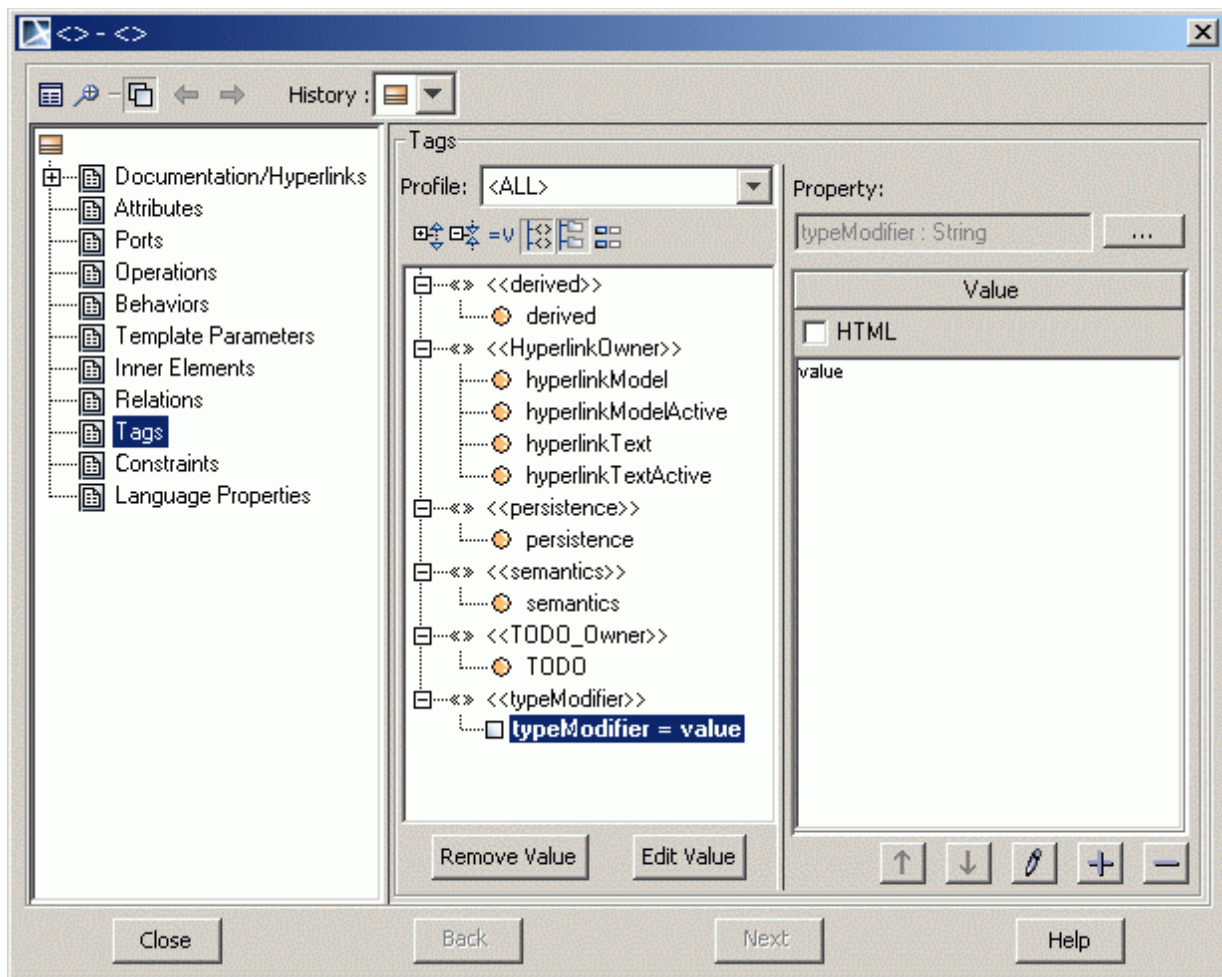







Figure 352 -- Tags dialog box

Button	Description
	Moves the created value to an upper position in the list.
Up	
	Moves the created value to a lower position in the list.
Down	
	Opens an editable window for value.
Edit	
	Adds a new tagged value to the list.
Add	
	Removes a tagged value from the list.
Remove	

To create default tag values

1. In the **Stereotype** specification dialog box, expand the **Tag Definitions** branch and select the tag definition.
2. Specify a default value in the **Default Value** field.
3. Create an element and assign it a stereotype. The element will have tags with assigned default values.

To set a default tag value to tag with empty value

MagicDraw version 15.0 and later allows setting default tag values to tag with empty value. This functionality is needed when a stereotype is already assigned to an element and a new mandatory tag definition with a default value is created for the stereotype. After creating such a tag definition, the model elements that have the modified stereotype applied will have the newly created tags unset.

To set default values instead of empty values:

1. From the **Tools** main menu, select the **Set Empty Tags to Defaults** command. The **Select Package** dialog box opens.
2. Select the scope of elements to which you want to set the default tag values.
Note If you want to assign default values to all project tags, in the *All data tree* select the *Data* model and click the **Add All** button. *Data* package is added to the *Selected objects* list. Click OK.
3. The **Question** dialog box opens informing you that this action will set the mandatory tags without values to defaults.
If you do not want to see this message again next time, clear the **Show this message next time** check box.
4. Click **Yes**. Now, the element with stereotype tags has been assigned default values.

The following conditions are required to set the default tag values:

1. The element should have an assigned stereotype with a specified default value property.
2. The tag definition, to which you want to assign a default value, should contain no value.

NOTE The default tag values are not set when the stereotype property (tag definition) multiplicity is equal to 0.

Parent Topic: “UML Extension Elements” on page 802.

Related topics

“Stereotype” on page 803.

“Attribute” on page 849.

Constraint

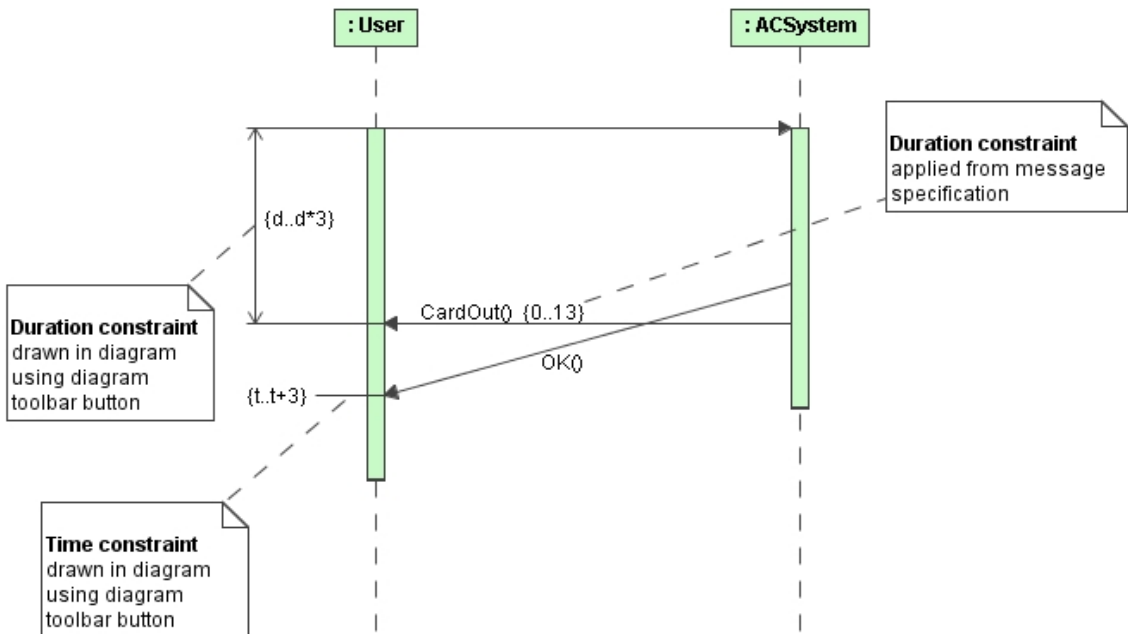
A Constraint represents additional semantic information attached to the constrained elements. It is an assertion that indicates a restriction that must be satisfied by a correct design of the system. The constrained elements are those elements required to evaluate the constraint specification. In addition, the context of the Constraint may be accessed, and may be used as the namespace for interpreting names used in the specification.

A Constraint is represented as a linguistic, enclosed in braces (`{}`), statement in some formal (OCL, C++, and other), or a natural language. There are 14 standard constraints in UML such as association, global, and parameter. You may also define your own constraints.

The Time and Duration Constraints

The Time Constraint specifies the combination of min and max timing interval values.

The Duration Constraint defines a value specification that specifies the temporal distance between two time instants.



Working with Constraints

Edit constraints in the Constraints tab of the Model Element Specification dialog box.

To define a new constraint

- From the element shortcut menu in the Browser, select **New**, and then select **Constraint**.
- Click the **Inner Elements** tab in the **Specification** dialog box for each model element and click the **Create** button (select **Constraint** from the open list, if needed). Specify the information about the constraint in the open **Constraint Specification** dialog box.
- Click the **Constraints** tab in the **Specification** dialog box for each model element and click the **Create** button. Type the name and specification of the constraint.

NOTE

If a constraint is displayed in the **Constraints** tab of the Element Specification dialog box, it means this constraint is valid and applied for the element. If it is displayed in the **Inner Elements** list, this constraint is only owned by the element.

To apply a constraint to an element

1. Click the **Constraints** tab in the **Specification** dialog box for each model element and click the **Apply** button.
2. The **Select Elements** dialog box opens. Select a constraint existing in the model from the **All Data** tree and click the **Add** button to move it to the **Selected Objects** list.
3. Click **OK**.

The Constraint Specification dialog box

To open the **Constraint Specification** dialog box

In the **Edit Constraints** dialog box, click the **Add** or **Edit** button.

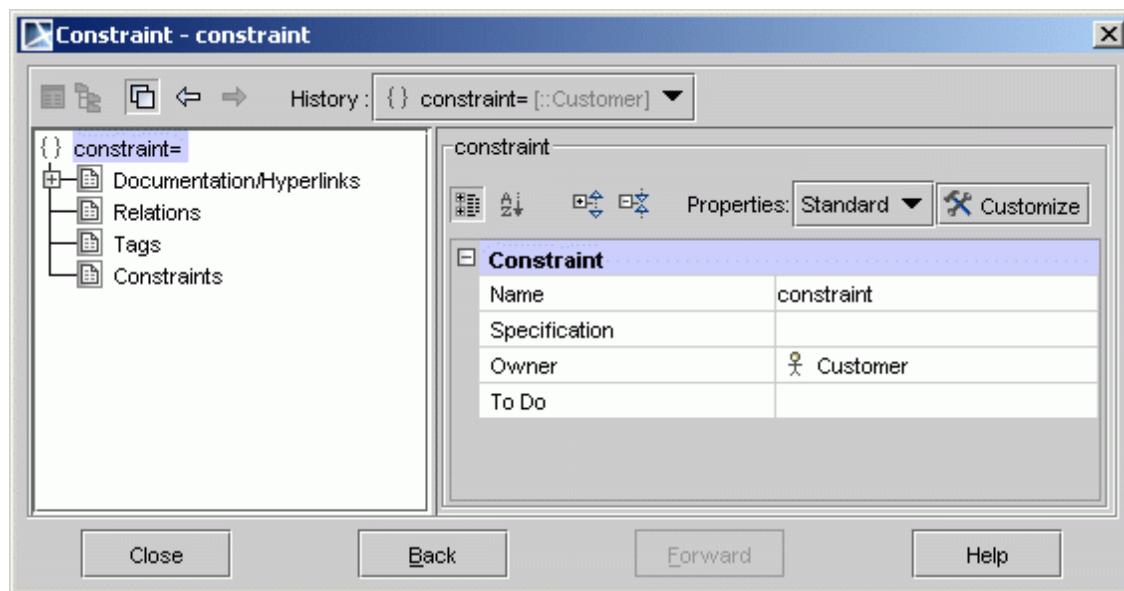


Figure 353 -- Constraint Specification dialog box

Tab name	Box	Function
----------	-----	----------

General Set a general information about the constraint.	Name	Enter the name of the constraint.
	Owner	The name of the model element to which the current constraint is assigned.
	Specification	Enter an expression of the constraint. Click “...” button and in the Edit Specification window, choose OCL. The specification will be checked according to Object Constraint Language (OCL)
	ToDo	Additional information can be added.
Documentation/ Hyperlinks	Documentation	Set the tagged value documentation. It can be HTML text also.
	Hyperlink	Choose a hyperlink for the tagged value. It can be a file, model element, or web site.
Tags		Creates, edits, and removes tagged definitions.
Constraints		Creates, applies, and removes a constraint for a tag.

OCL

Object Constraint Language (OCL) is a formal language used to express constraints. These typically specify the invariant conditions that must hold for the system being modeled.

Expressions can be used in a number of places in a UML model:

- To specify the initial value of an attribute or association end.
- To specify the derivation rule for an attribute or association end.
- To specify the body of an operation.
- To indicate an instance in a dynamic diagram.
- To indicate a condition in a dynamic diagram.
- To indicate the actual parameter values in a dynamic diagram.

There are four types of constraints:

- An *invariant* is a constraint that states a condition that must always be met by all instances of the class, type, or interface. The invariant is described using an expression that evaluates to true if the invariant is met. Invariants must be true all the time.
- A *precondition* to an operation is a restriction that must be true at the moment the operation is going to be executed. The obligations are specified by the postconditions.
- A *postcondition* to an operation is a restriction that must be true at the moment the operation has just been executed.
- A *guard* is a constraint that must be true before a state transition discharged.

Invariants on attributes

The simplest constraint is an invariant on an attribute. Suppose a model contains a class *Customer* with an attribute *age*, then the following constraint restricts the value of the attribute:

```
context Customer inv:  
age >= 18
```

Invariants on associations

One may also put constraints on the associated objects. Suppose a model contains the class *Customer* that has an association to the class *Salesperson* with the role name *salesrep* and multiplicity 1, then the following constraint restricts the value of the attribute *knowledgelevel* of the associated instance of *Salesperson*:

```
context Customer inv:  
salesrep.knowledgelevel >= 5
```

Collections of objects

In most cases the multiplicity of an association is not 1, but more than 1. Evaluating a constraint in these cases will result in a collection of instances of the associated class. Constraints can be put on either the collection itself, e.g. limiting the size, or on the elements of the collection. Suppose in a model the association between *Salesperson* and *Customer* has the role name *clients* and multiplicity 1..* on the side of the *Customer* class, then we might restrict this relationship by the following constraints:

```
context Salesperson inv:  
clients->size() <= 100 and clients->forall(c: Customer | c.age >= 40)
```


Pre- and postconditions

In the pre- and postconditions the parameters of the operation may be used. Furthermore, there is a special keyword `result` which denotes the return value of the operation. It can be used in the postcondition only. For example an operation *sell* was added to the *Salesperson* class.

```
context Salesperson::sell( item: Thing ): Real
pre: self.sellableItems->includes( item )
post: not self.sellableItems->includes( item ) and result = item.price
```

Derivation Rules

Models often define derived attributes and associations. A derived element does not stand alone. The value of a derived element must always be determined from other (base) values in the model. Omitting the way to derive the element value results in an incomplete model. Using OCL, the derivation can be expressed in a derivation rule. In the following example, the value of a derived element *usedServices* is defined to be all services that have generated transactions on the account:

```
context LoyaltyAccount::usedServices : Set(Services)
derive: transactions.service->asSet()
```

Initial Values

In the model information, the initial value of an attribute or association role can be specified by an OCL expression. In the following examples, the initial value for the attribute *points* is 0, and for the association end *transactions*, it is an empty set:

```
context LoyaltyAccount::points : Integer
init: 0
context LoyaltyAccount::transactions : Set(Transaction)
init: Set{}
```

Body of Query Operations

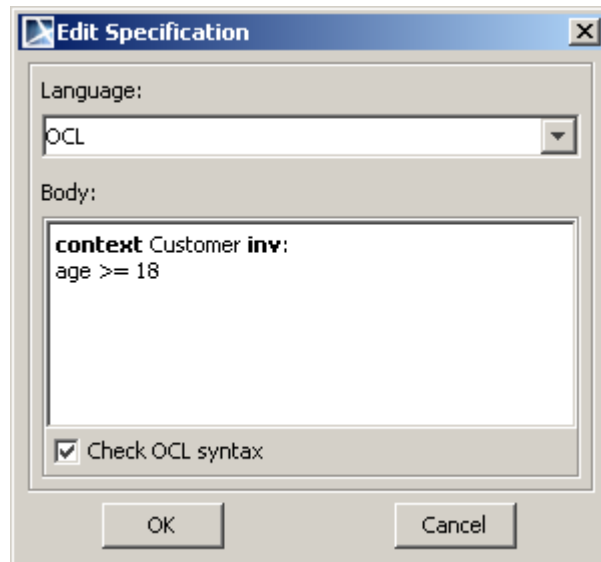
The class diagram can introduce a number of query operations. The query operations are operations that have no side effects, i.e. do not change the state of any instance in the system. The execution of a query operation results in a value or set of values without any alterations in the state of the system. The query operations can be introduced in the class diagram, but can only be fully defined by specifying the result of the operation. Using OCL, the result can be given in a single expression, called a body expression. In fact, OCL is a full query language, comparable to SQL. The use of body expressions is an illustration thereof.

The next example states that the operation *getCustomerName* will always result in the name of the card owner associated with the loyalty account:

```
context LoyaltyAccount::getCustomerName() : String  
body: Membership.card.owner.name
```

To check OCL syntax according to OCL grammar

1. Click the “...” button near the **Specification** field in the **Constraint Specification** dialog box. The **Edit Specification** dialog box opens.
2. Select the **OCL** in **Language** field and check the **Check OCL syntax** check box. Incorrect expression in the **Body** text field will be highlighted in red.



Profiles

A Profile is a kind of Package that extends a reference metamodel. The primary extension construct is the Stereotype, which are defined as part of the Profiles.

A profile introduces several constraints, or restrictions, on ordinary metamodeling through the use of the metaclasses defined in the package. It is a restricted form of a metamodel that must always be related to a reference metamodel, such as UML, as described below. It cannot be used without its ref-

erence metamodel, and it defines a limited capability to extend metaclasses of the reference metamodel. The extensions are defined as stereotypes that apply to the existing metaclasses.

The profile is defined as a package – it has package properties – in the **Profile Specification** dialog box. For a detailed description of packages, see “Package” on page 955.

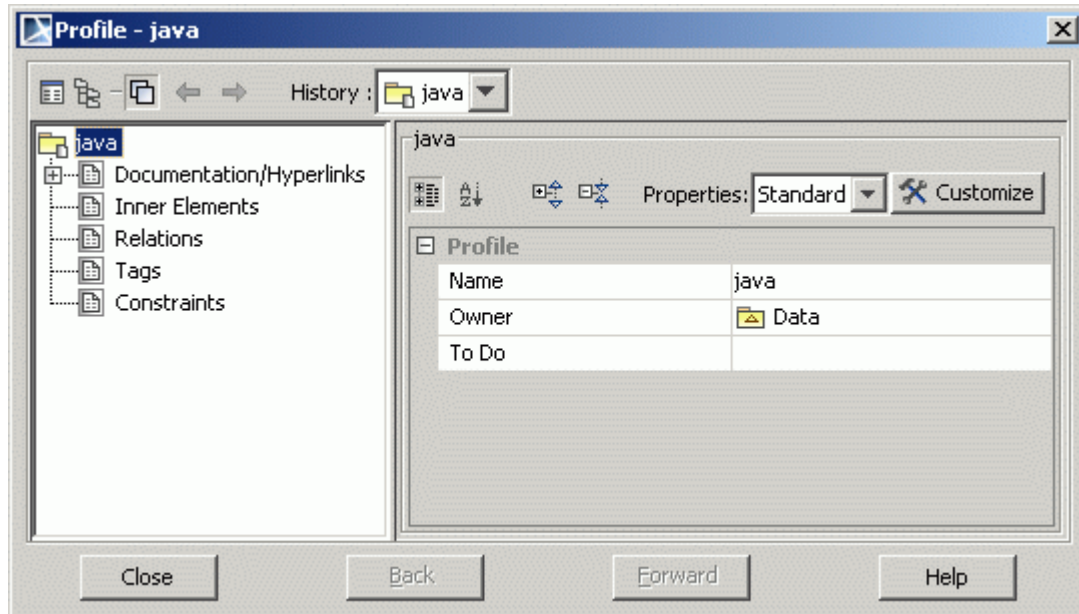


Figure 354 -- Subsystem Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Profiles

The Profiles are defined using the UML extensibility mechanisms. These allow modelers to customize UML for specific domains such as software development processes. In MagicDraw, the mechanism of the profile is similar to the functionality of the modules, except that the profile is a package containing the stereotype <<profile>>, and its content is visible in both the Profiles dialog box and the UML extensions tabs of the Specification dialog boxes.

Working with profiles

There are several ways to work with profiles. You may:

1. Create your own profile.
2. Use the preexisting one.
3. Import the preexisting one.
4. Export your created profile.

Further details on these functions can be found in the following sections.

Creating Profiles

Usually a profile contains specific stereotypes, constraints, and tagged values that you can use according to your needs. The Profiles often contain other specific model elements, which MagicDraw allows you to include in your profile. You can use your existing profile in other parts of the projects. Additionally, other users may reuse your profile as a module.

To create your own profile

1. Create a profile element (in the Browser or using the Class Diagram toolbar, Profiling Mechanism group)
2. Add the stereotypes, constraints, and/or other model elements you need for extending UML.

Using and Importing Profiles

You may use other profiles in your project, or import them. MagicDraw offers several profiles, found in the <MagicDraw installation directory>/profiles folder.

To use the preexisting profile in your project

1. From the **File** menu, select **Use Module**.
2. The **Use Module** dialog box opens. If needed, select the directory where the desired profile is located. Select the desired profile, and click **OK**.

The selected profile opens in the Browser tree as a read-only profile. You may use it in your project. You can also import and edit it, if necessary.

To import the preexisting profile into your project

- From your selected profile, go to the Browser's shortcut menu. Select **Modules** and then **Import Module**.
- From the **File** menu, select **Import**. The **Import** dialog box opens. Select the profile you want to import and click **Open**.

The imported profile will open as an editable package in the Browser.

Exporting Profiles

If you want to use the existing profile in another project, or enable other users to use it, you need to save it in your selected directory. Exporting profiles is similar to exporting module functionality.

1. From the **File** menu, select **Export**, and then **Module**.

The **Export Module** dialog box opens. Select the profile package you want to export and add it to the **Selected** list. Write the description of the profile in the text box, click **OK**, and save it to the desired directory.

Action

An action is a named element that is the fundamental unit of an executable functionality. The execution of an action represents some transformations or processing in the modeled system, be it a computer system or otherwise. An action execution represents the run-time behavior of executing an action within a specific behavior execution. As action is an abstract class, all action executions will be the executions of a specific kind of action. When the action will be executed and what its actual inputs will be are determined by the concrete action and the behaviors in which it is used.

The following are types of actions:

- “Call Behavior Action” on page 828.
- “Call Operation Action” on page 829.
- “Opaque Action” on page 831.
- “Send Signal Action” on page 831
- Any other Action.

See also: “Working with actions” on page 832.

Usage in diagrams: “Activity Diagram” on page 685.

Parent topic: “Model Elements” on page 796.

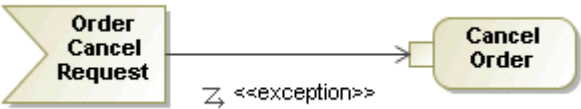
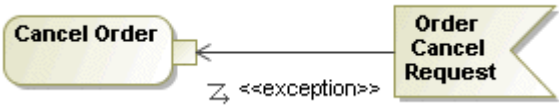
Related topics

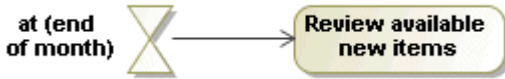
“Specification dialog boxes” on page 325.

“Formatting Symbols” on page 342.

Accept Event Action

The Accept event action is an action that waits for the occurrence of an event that meets specified conditions. If an accept event action has no incoming edges, then the action starts when the containing activity or structured node starts, whichever most immediately contains the action. In addition, an accept event action with no incoming edges remains enabled after it accepts an event. It does not terminate after accepting an event and outputting a value, but continues to wait for other events. An accept event action with no incoming edges and contained by a structured node is terminated when its container is terminated.

Notation	Description
<p>Accept event action</p> 	<p>The <i>Order cancel request</i> accept event action is connected to <i>Cancel order</i> with the exception handler relationship.</p>
<p>Rotated accept event action</p> 	<p>The symbol of accept event action is rotated.</p>

Accept time event action

The time event is connected to the call behavior action.

To set a trigger to an accept event action

1. Double-click the accept event action shape or select **Specification** from the shape shortcut menu.
2. Click "+" in the **Trigger** field. The **Trigger** specification dialog box opens. Specify the trigger, which will be assigned for the accept event action.


To draw the time event

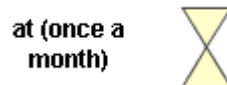
To draw the accept event action with the time event shape, in the activity diagram toolbar, select the **Time Event** button and then click on the diagram pane.

To change the position of the accept event action name

1. Select the accept event action on the diagram pane:



2. Click the **Rotate State** button  on the upper-right side of the shape. Now the accept event action shape looks like the following:



To specify “Is Unmarshal” property

In the **Accept Event Action** specification dialog box, select/clear the **Is Unmarshal** property, which indicates whether there is a single output pin for the event, or multiple output pins for the attributes of the event.

Call Behavior Action

The call behavior action invokes a behavior directly rather than invoking a behavioral feature that, in turn, causes the behavior. The argument values of the action are available to the execution of the invoked behavior. The execution of the call behavior action waits until the execution of the invoked behavior completes and a result is returned on its output pin. In particular, the invoked behavior may be an activity.



The Receive Order and Fill Order call behavior actions

To assign a behavior to the call behavior action

- From the call behavior action shortcut menu select the **Behavior** command.
- In the **Call Behavior Action** specification dialog box, modify the **Behavior** field.

NOTES

- Double click the call behavior action with the assigned behavior - the behavior specification dialog box is opened or if the assigned behavior is a diagram, the diagram will be opened.
- After the behavior is assigned to the call behavior action, a rake symbol is displayed on the action shape.

: Production testing
m

To create a new diagram for the call behavior action

From the call behavior action symbol's shortcut menu, select **New Diagram** and then diagram from the list. This accelerates the creation of behavior diagrams.

To select the name display mode on the call behavior action

Stereotypes from the behavior can be visible on the call behavior action. You can select to show an action name, a behavior name or both by changing the **Name Display Mode** property in the call behavior action symbol **Properties** dialog box.

Call Operation Action

The call operation action transmits an operation call request to the target object, where it may cause the invocation of an associated behavior.

You can display an action name and/or name of the operation on the call operation action shape. For example, if you have two call operation action elements calling the same operation, you may specify their names to distinguish which action means what.

Notation	Description
<p>The call operation action with <code>getOrder</code> operation</p> <p>A yellow rounded rectangle containing the text <code>getOrder</code> in bold and <code>(OrdersDB::)</code> in parentheses below it.</p>	<p>In this example the call operation action has been assigned <i>getOrder</i> operation whose type is <i>OrdersDB</i> class.</p>
<p>The call operation action named <i>Get supplementary order</i></p> <p>A yellow rounded rectangle containing the text <i>Get supplementary order</i> in bold and <code>(OrdersDB::getOrder)</code> in parentheses below it.</p> <p>order</p>	<p>In this example the call operation action is named <i>Get supplementary order</i>. It has been assigned <i>getOrder</i> operation whose type is <i>OrdersDB</i> class.</p>
<p>Call operation action with hidden classified name</p> <p>A yellow rounded rectangle containing the text <code>getOrder</code> in bold.</p>	<p>The classified name of operation is hidden from the call operation action shape.</p>

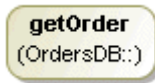
To assign an operation for the call operation action

From the action shortcut menu, select the **Operation** command or set an operation in the **Call Operation Action** specification dialog box, **Operation** field.

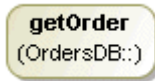
MagicDraw version 15.0 and later allows the display of operation name and class of operation on the call operation action shape.

When an operation is assigned to the call operation action, there are three name/operation display options available:

- If the call operation action is not named, the name of the class is displayed under the operation name.



- If the name of the call operation action is the same as the assigned operation name, then the name of the class is displayed under the operation name.



- If the call operation action name differs from the assigned operation name, then *<class of the operation>::<operation name>* is displayed under the call operation action name.



To hide the operation name and class of the operation from the call operation action shape

From the call operation action shortcut menu, clear the **Show Qualified Name for Operation** check box. You may also customize this property in the call operation action **Properties** dialog box.

NOTE

When loading a project from a version older than MagicDraw version 15.0, by default the class of the operation is not displayed on the diagram pane. The **Show Qualified Name for Operation** option is unchecked.

Opaque Action


The opaque action is introduced for implementation-specific actions or for use as a temporary placeholder before some other actions are chosen.

The opaque action has no specific notation.

There are additional **Body** and **Language** text fields in the **Opaque Action** specification dialog box.

Send Signal Action

The send signal actions creates a signal instance from its inputs and transmits it to the target object, where it may cause the start of a state machine transition or the execution of an activity. The argument values are available to the execution of associated behaviors. The requester continues the execution immediately. Any reply message is ignored and is not transmitted to the requester. If the input is already a signal instance, use the Send object action.

Notation	Description
<p style="text-align: center;">Send signal action</p>  <pre> graph LR A(Create order) --> B(Fill order request) B --> C(Create invoice) </pre>	<p>This example describes an order process.</p> <ol style="list-style-type: none"> 1. First, an order is created (<i>Create order</i> call behavior action). 2. Then, a signal to fill the order request is sent to the warehouse (<i>Fill order request</i> send a signal action). 3. Finally, an invoice is created (<i>Create invoice</i> call behavior action). <p>The relationships are represented with control flow paths.</p>

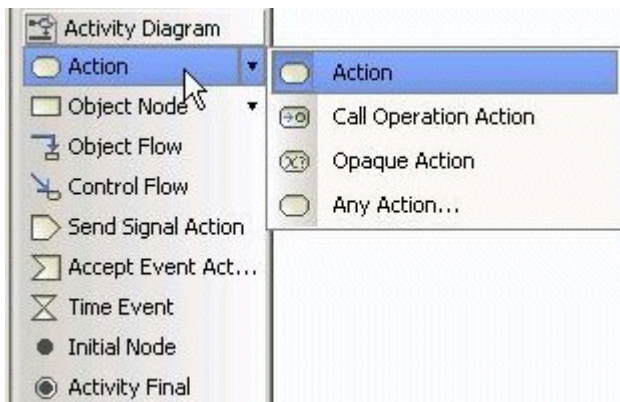
To assign a signal to the send signal action

1. Double-click the send signal action shape or select **Specification** from the shape shortcut menu. The **Send Signal Action** specification dialog box opens.
2. Click “...” in the **Signal** Field. The **Select Element** dialog box opens.
3. Select the signal or create a new one.

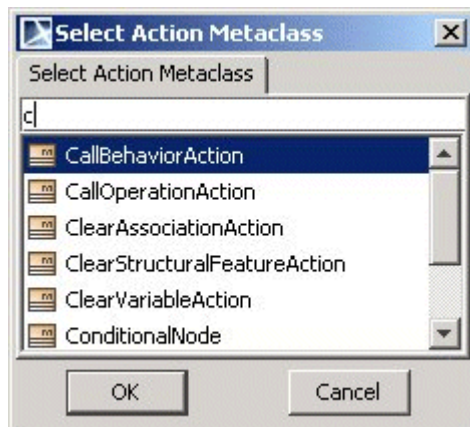
Working with actions

To quickly create any action

1. In the activity diagram toolbar, right-click the **Action** button. The menu opens.



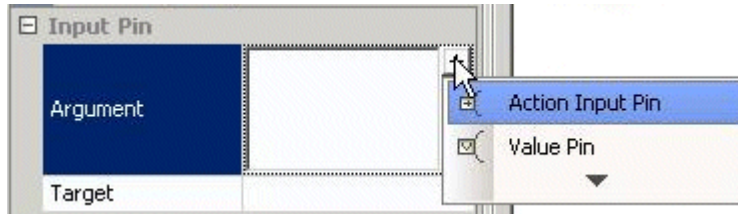
2. Select the **Any Action** command. The **Select Action Metaclass** dialog box opens.



3. Select an action metaclass from the list or type the first letter of the metaclass. Click **OK**. The action is created.
4. Click on the diagram pane. An action symbol is drawn.

To assign pins for an action

1. In the action specification dialog box, select the **Pins** branch.
2. In the right window size, in the **Input Pin** field, select the **Argument** box. Click '+' button. A menu with input pin, action input pin, or value pin opens.



3. Select the pin from the list. The pin specification dialog box opens.
 - In the **Output Pin** field, select the **Result** box and click '+' button. The **Output Pin** specification dialog box opens.

NOTES

- The Output pin is not included when selecting the send signal action.
- The Input pin is not included when selecting the accept event action.

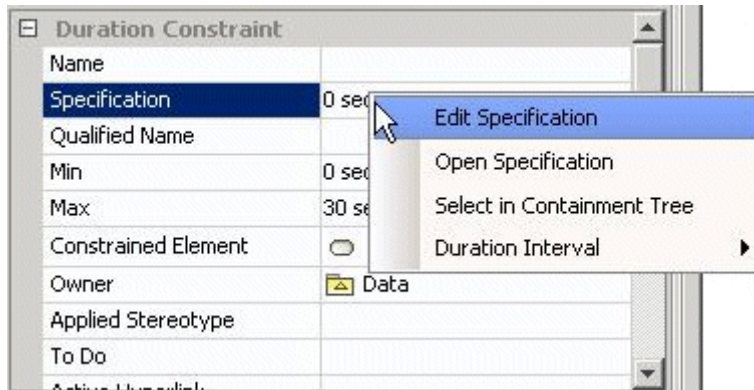
Advancing actions: applying duration constraint

You can create and apply a time duration constraint on an action that states that the output must occur after delay from the input.

To create and apply a time duration constraint

1. Select an action and create the input and output pins to specify the events.
2. Create a duration constraint for the action:
 - a In the **Call Behavior Action** specification dialog box, select the **Constraints** branch. At the bottom of the right side pane, click the **Apply** button. The **Select Elements** dialog box opens.
 - m Select the constraint storage place and click the **Create** button. In the menu that opens, select the **Duration Constraint** command. The **Duration Constraint** specification dialog box opens.
3. Specify a duration interval. In the **Duration Constraint** dialog box, select the **Expert** property display mode. Type the minimum and maximum duration for holding an activity in the **Min** and **Max** fields, e.g. type the following values: 0 sec and 30 sec.
4. Assign the events for the input and output pins:

- a In the open **Duration Constraint** specification dialog box, right-click the **Specification** field. The following menu opens:

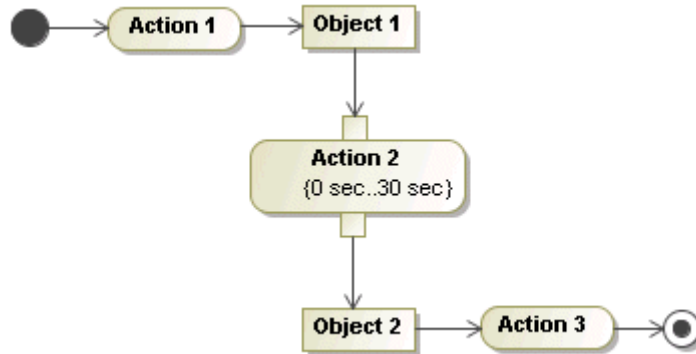


- b Select the **Open Specification** command. The **Duration Interval** specification dialog box opens.
- c Right-click the **Min** property and select the **Open Specification** command. The **Duration** specification dialog box opens.
- d In the **Event** field, click the "... " button. The **Select Element** dialog box opens. Select the activity input pin.
- e Repeat steps c and d for the **Max** property - select the action output pin as an event.
5. Apply the created duration constraint on the action.

TIP

To return to the former dialog box in the specification dialog box, click the **Back** button.

The sample below depicts an activity diagram with a duration constraint applied on an action.



Actor

Actors represent roles played by human users, external hardware, and other subjects. An actor does not necessarily represent a specific physical entity but merely a particular facet (that is, "role") of some entities that is relevant to the specification of its associated use cases.

An actor requires task solutions from a system. This task is represented as a use case.

An actor is shown as a "stick man" figure with the name below the figure.

A general information about working with shapes is offered in Chapter 5, "Working With Diagrams."

Define an actor in the **Actor Specification** dialog box.

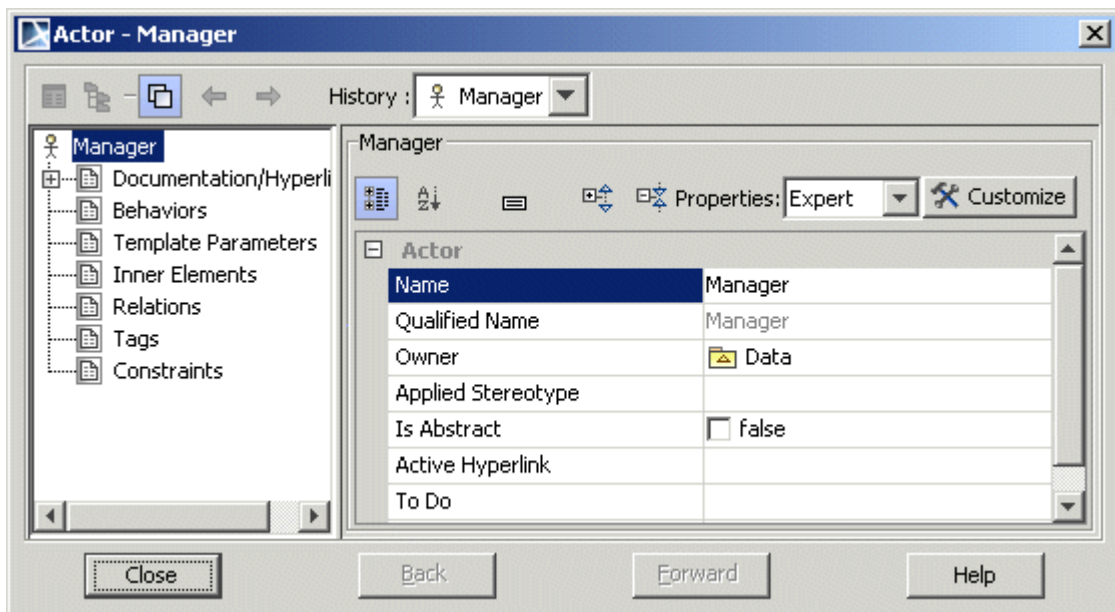


Figure 355 -- the Actor Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for more information on the specification elements.

Tab name	Box	Function
Inner Elements Add a constraint to a actor	Name	The constraint name.
	Type	The constraint type.
	Create	The Constraint Specification dialog box opens. Define the constraint in the dialog box.
	Delete	Removes the selected constraint element from the actor.

To define an actor as abstract

1. Double-click the selected actor or select **Specification** from the actor shortcut menu. The **Actor Specification** dialog box opens.

2. Select the **Is Abstract** check box in the general properties group.

To draw an actor icon in the sequence diagram

1. Drag an actor from the Browser and drop it on the diagram pane.
2. From the classifier shortcut menu, select the **Suppress Content** command.

Association

An association in the class diagrams represents the semantic relationship between two or more classifiers, which specifies connections between their instances. An association relationship is the most general of all relationships and the most semantically weak.

An association in the use case diagrams represents the participation of an actor in a use case, i.e., when instances of the actor and instances of the use case communicate with each other. This is the only relationship between actors and use cases. Sometimes an association relationship is called communication association.

An association is drawn as a solid path connecting two classifier symbols.

For a general information about working with symbols, see Chapter 5, "Working With Diagrams."

Specify an association in the Association Specification dialog box.

Refer to the "Specification dialog boxes" on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set a general information about the association.	Association End A	
	Name	The name of the class A. Click the '...' button, to open the Name dialog box.
	Navigable	If selected, a navigable owned end of the association indicates that the association is navigable from the opposite ends, otherwise the association is not navigable from the opposite ends.

Tab name	Box	Function
	Owned By	<ul style="list-style-type: none"> If a Classifier is selected, the classifier name from the opposite association end is displayed on the association end. If an Association is selected, the name of the association to which the association end belongs is displayed on the association end.
	Multiplicity	The multiplicity value of an association end A.
	Default Value	Click the '...' button to open the Select Elements dialog box and select the value for the model elements tree.
	Association End B	
	Name	The name of the association end B. Click the '...' button, to open the Name dialog box.
	Navigable	If selected, a navigable owned end of the association indicates that the association is navigable from the opposite ends, otherwise the association is not navigable from the opposite ends.
	Owned By	<ul style="list-style-type: none"> If a Classifier is selected, the classifier name from the opposite association end is displayed on the association end. If an Association is selected, the name of the association to which the association end belongs is displayed on the association end.
	Multiplicity	The multiplicity value of an association end B.
	Default Value	Click the '...' button to open the Select Elements dialog box and select the value for the model elements tree.

To show the direction arrow near the association name

From the association shortcut menu, select the **Show Direction Arrow** check box.

Default Direction Arrow direction is displayed according path creation direction. To change the Direction Arrow direction from the association shortcut menu, select the **Reverse Direction Arrow** command.

The Direction Arrow is graphical display most often used in top level domain class diagrams. The Direction Arrow helps to read diagram and explain diagram semantics. The Direction Arrow has no meaning in a model.

Usually Direction Arrow is used in diagram where navigability is not defined yet. Direction Arrows are usually displayed for named associations. When you move on with your modeling and create more detail diagrams with specified navigability, direction arrows and associations names usually are not displayed in this type of diagrams.

See an example in Figure 356 on page 839 and Figure 357 on page 839. *User* and *Account* classes are connected with association. Navigation arrow may be displayed to either side, depending on the association name. If association name is “*belongs to*” - Direction Arrow should point from *Account* class to *User* class. If association name is “*has*” - Direction Arrow should point from *User* class to *Account* class.

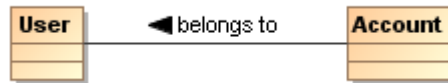


Figure 356 -- The Direction Arrow points from Account class to User class

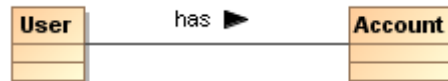


Figure 357 -- The Direction Arrow points from User class to Account class

To draw an association class

In the class diagram, you may add attributes to an association using an association class. The association class is a simple class that has a dashed line connected to the association.

1. Draw two classes.
2. Click the **Association Class** button on the diagram toolbar.
3. On the diagram pane click the first class shape (path source).

4. Drag the path to the second class (path destination) and drop it there.

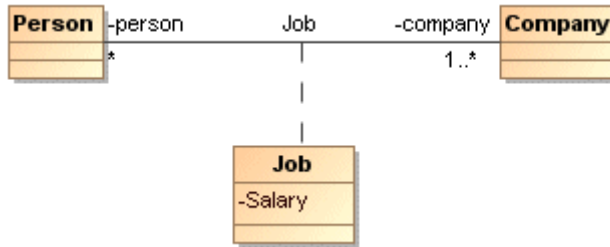


Figure 358 -- Sample of the association class

NOTE If you need to model a relationship among a number of classes, N-ary association is used.

To draw an N-ary association class

The N-ary association is drawn as a big diamond with all the associations attached to its points. Every involved class may have a role name and multiplicity.

1. Draw three classes on the Diagram pane.
2. Draw the N-ary association connector icon.
3. Connect all classes and the N-ary association connector icon using an association path.



Adding Association between Read-only Classifiers

Adding new Association always creates two roles or properties at both ends that are owned by the attached Classifier by default. However, when one or both ends of the Association is or are not editable for some reasons (for example, locked in Teamwork Server or located in a read-only profile/module), the properties will be owned by the Association itself. In this case, MagicDraw will display a

warning informing you about the sometimes-unexpected issue of model creation (Figure 359 on page 841, Figure 360 on page 841).

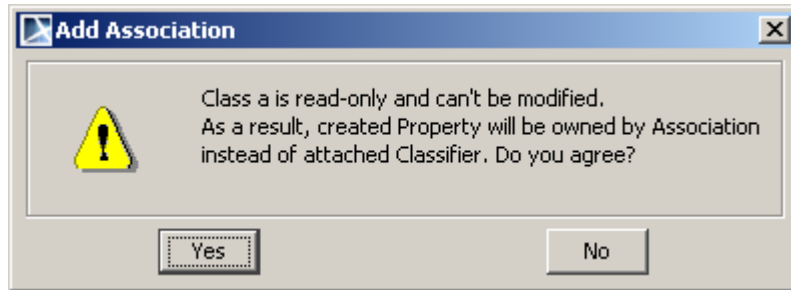


Figure 359 -- The Add Association Dialog for a Read-Only Classifier

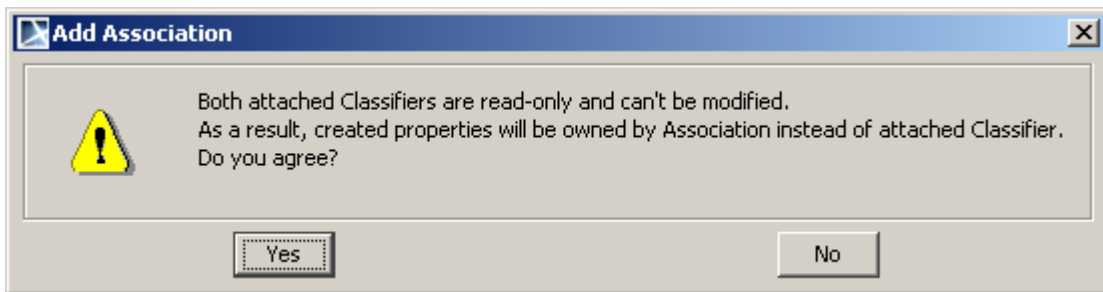


Figure 360 -- The Add Association Dialog for Both Read-Only Classifiers

Association End

The associations ends are represented by properties, each of which is connected to the type of the end. When a property is an association end, the value or values are related to the instance or instances at the other end(s) of the association.

An association end is the connection between the lines depicting an association and the icon (often a box) depicting the connected classifier.

To open the **Association End Specification** dialog box

1. Double-click the association path. The **Association Specification** dialog box appears.
2. Select the **Association Ends** group in the specification tree and expand it by clicking the plus sign.
3. Double click the selected **Association End**. The **Property Specification** dialog box opens.
 - From the association end shortcut menu, select **Specification**.

An association end is defined as a property – it has attribute properties in the **Property Specification** dialog box. For a detailed description of attributes, see “Drag and drop the selected Opaque Behavior element from the Browser tree on the Diagram pane.” on page 947.

Click the **Expert** in the **Properties** field for showing more properties of the association end.

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set general information about the association end	Type	Shows an association end type - name of the class, to which the association end is related.
	Default Value	Click the ‘...’ button to open the Select Elements dialog box and select the value for the model elements tree.
	Owner	Shows a class name, containing an association end as the property.
	Type Modifier	Additional information about the type. <ul style="list-style-type: none"> • & - one class has a reference to other model elements. • * - one class has a pointer to other model elements. • [] – one class has an array of other model elements.
	Navigable	If selected, a navigable owned end of the association indicates that the association is navigable from the opposite ends, otherwise the association is not navigable from the opposite ends.
	Multiplicity	The multiplicity value of an association end A.
	Is Read Only	If <i>true</i> , the association end may only be read, not written.

Tab name	Box	Function
	Is Static	This property scope means that the values returned by the property have no duplicates.
	Aggregation	<p>When placed on a target end, specifies whether the target end is an aggregation with respect to the source end. Only one end can be an aggregation.</p> <ul style="list-style-type: none"> • None. The end is not an aggregate. • Shared. The end is an aggregate; therefore, the other end is a part and must have the aggregation value of none. The part can be contained in other aggregates. • Composite. The end is a composite; therefore, the other end is a part and must have the aggregation value of none. The part is strongly owned by the composite and may not be part of any other composites.
	Is Derived	Specifies whether the Property is derived, i.e., whether its value or values can be computed from other information.
	Is Ordered	<p>If the multiplicity is greater than one, then the set of related elements can be ordered or unordered.</p> <ul style="list-style-type: none"> • <i>False</i> - the elements form an unordered set. This is the default and need not be shown explicitly. • <i>True</i> - the elements of the set are ordered, but duplicates are still prohibited. This generic specification includes all ordering types.
	Is Unique	If <i>true</i> , the collection of values may not contain duplicates.
Qualifiers Define qualifiers of the association end. A qualifier is an attribute of an association end.	Name	The name of the assigned attribute of an association end.
	Type	The type of the assigned attribute of an association end.
	Property	Shows the owner of the qualifier - to which property it belongs.
	Default value	The initial value of an attribute of an association end.
	Up	Moves item to an upper position in the list

Tab name	Box	Function
	Down	Moves item to a lower position in the list.
	Create	The Property Specification dialog box opens.
	Delete	Removes the qualifier.

To define an association end name

1. Open the **Property Specification** dialog box.
2. Type an association end name in the **Name** box.
 - Select **Role A of** <class name> or **Role B of** <class name> from the association shortcut menu, and then select the **Edit Name** subcommand, then type or edit the name directly on the Diagram pane.

To make a special case of an association path (aggregation/composition)

1. Open the **Property Specification** dialog box.
2. Click the desired option button for an aggregation kind (**None**, **Shared** or **Composite**).
 - From the association shortcut menu, select **Role A of** (class name) or **Role B of** (class name), and then select **None**, **Shared**, or **Composite**.
 - Click the **Composition** or **Aggregation** button and draw an appropriate path on the diagram.
 - Right click the association path end and select **Shared** or **Composite** command from the shortcut menu.

To change the association navigability

The association navigability indicates whether it is possible to traverse an association within an expression of a classifier to obtain the object or set of objects associated with the instances. The navigability is shown as an arrow that can be attached to the end of the path to indicate that the navigation is supported toward the classifier attached to the arrow.

- Open the **Association End Specification** dialog box and select/clear the **Navigable** check box.

- Select **Role A of** (class name) or **Role B of** (class name) from the association shortcut menu, and then select/clear the **Navigable** check box..

NOTE

By default, an association is navigable on both sides and its navigability is not visible.

To display the association navigability

From both association ends shortcut menu, select the **Show Navigability** check boxes.

In the Figure below the association is navigable on both sides and its navigability is visible.

**NOTES**

- To open the association ends shortcut menu, right-click on the end of the association (on the association end area) on the diagram pane.
- Open the association shortcut menu and select the Role A of <class name> or Role B of <class name>.

To define the association end visibility

A role indicates a role played by the class in terms of an association. The role name is placed at the association end, near the class playing that role. The role name at the implementation level maps into the reference name to the opposite class. Roles may have visibility (public, package, protected, and private).

1. Open the **Property Specification** dialog box.
2. From the **Visibility** drop-down list box, choose **public**, **package**, **protected**, or **private**.

To place multiplicity values in the association path ends

The association end multiplicity describes how many entities are participating at each association end:

- 0 – zero and only zero.
- 1 – one and only one.

- 0..1 – zero or one.
- 0..* – from zero to any positive integer.
- 1..* – from one to any positive integer.
- * – any positive integer.

1. Open the **Property Specification** dialog box.
2. Type or select from the list multiplicity value (1, *, 0..*, etc.) in the **Multiplicity** box.
 - Open the **Association Specification** dialog box and from the **Multiplicity** drop-down list box, select or type the multiplicity value for the desired association end.
 - From the association shortcut menu, select **Role A of** (class name) or **Role B of** (class name), then select the multiplicity value (1, *, 0..*, etc.).

To add, edit, or remove a qualifier to/from an association end

A qualifier is an attribute or list of attributes whose values serve to partition the set of instances associated with an instance across an association. The qualifiers are attributes of the association. It is shown as a small rectangle attached to the end of an association path between the final path segment and the symbol of the classifier that it connects to. The qualifier rectangle is part of the association path, not part of the classifier. The qualifier rectangle drags with the path segments. The qualifier is attached to the source end of the association.

1. Open the **Association End Specification** dialog box, **Qualifiers** group.
2. Click the **Create** button. The **Property Specification** dialog box opens. Define a qualifier.
3. To remove the qualifier, click the **Delete** button.

To show the association ends as attributes on linked class shapes

If two classes are linked with an association path, both classes have a property created with an opposite class assigned as the property type in them. This property can be displayed on the class shape as well as an association link.

From the class shortcut menu, select **Symbol(s) Properties**. In the open **Properties** dialog box, the **Attribute** group, select one of the possible options near the **Show Association End** property:

- **All** - property and association paths will be displayed on the diagram pane.
- **Without Association Symbol** - if an association symbol is deleted, the property will be displayed on the class shape.
- **Do Not Show** - neither property, nor association path will be displayed on the diagram pane.

Advancing actions: navigable owned association ends

Navigability describes the need for an object to access another object by navigating across the link. According to the UML 2 specification, the association ends owned by the classes and associations are navigable. This improved functionality allows a proper management of the `navigableOwnedEnd` property for associations:

1. Able to manually change the ownership of an association end.

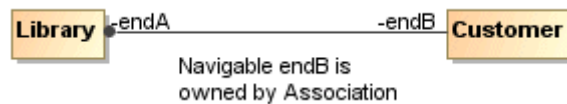


Figure 361 -- Sample of navigable endB, which is owned by association

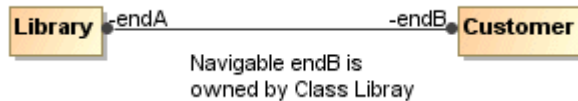


Figure 362 -- Sample of navigable endB, which is owned by Class Library

To change the ownership of an association end, select the **Owned By** command from the **Association End** shortcut menu and then select the desired owner.

2. Set the navigability for the association ends owned by the associations while keeping the ownership.

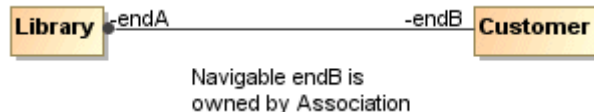


Figure 363 -- Sample of navigable endB, which is owned by Association

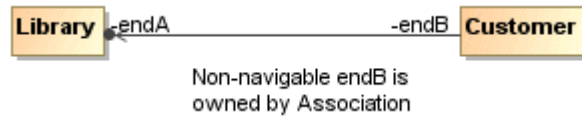


Figure 364 -- Sample of Non-navigable endB, which is owned by Association

3. Support the dot notation.

The ownership of association ends by an associated Classifier is now indicated graphically.

The following example shows that endA is owned by the Customer class and endB is owned by the association.

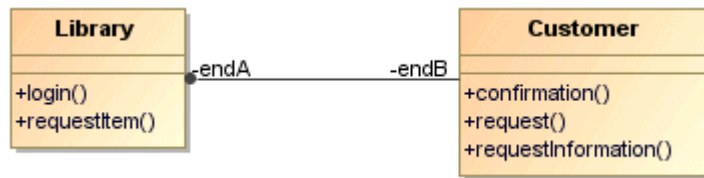


Figure 365 -- Sample of endA owned by class and endB owned by association

To enable the dot notation from the **Options** menu select the **Project** command and then select the **General project options** branch. Next, select the **Enable dot notation for associations** check box.

Association in Use Case Diagrams

The participation of an actor in a use case, i.e. instances where an actor and a use case communicate with each other. This is the only relationship between actors and use cases. The association relationships are also known as communication associations.

For more information on defining associations, see “Realization” on page 994.

Attribute

An attribute is a named property of a class that describes a range of values that can be held by the instances of that class.

To create a new attribute

- Double-click the selected class or select **Specification** from the class shortcut menu. The **Class Specification** dialog box opens. Click the **Attributes** tab and then click the **Create** button. The **Property Specification** dialog box opens. Define a new attribute and click **OK**.
- Select the **Insert New Attribute** from the class shortcut menu. Type the attribute name directly on the class shape.
- In the Browser tree, select an already created class. From the class item shortcut menu, select **New** and then **Property**.
- Press CTRL+ALT+A shortcut key and type the attribute name on the Diagram pane.
- Select a class shape and click the small orange **Insert New Attribute** smart manipulation button.

Define an attribute in the **Property Specification** dialog box.

To open the **Property Specification** dialog

1. Open the **Class Specification**, **Actor Specification**, or **Interface Specification** dialog box.
 2. In the **Attributes** tab, double-click the desired attribute in the tree, or click the **Create** button.
- Draw an association path and click on the association end.

- Double-click the desired attribute directly on the diagram.

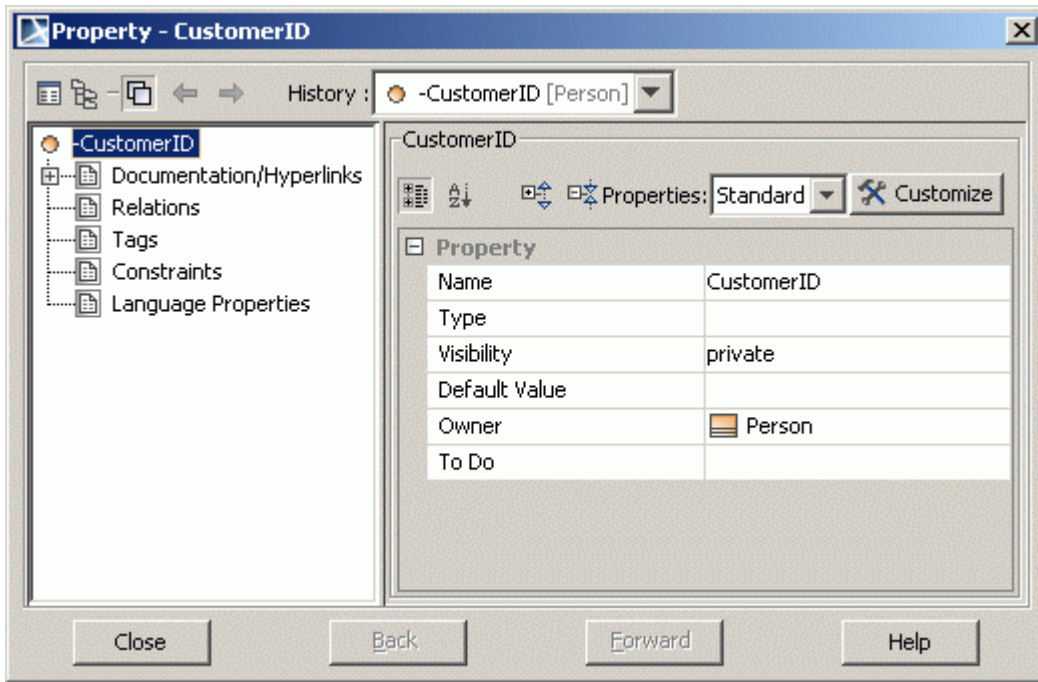


Figure 366 -- Property Specification dialog box

Click the **Expert** in the **Properties** field for showing more properties of the attribute.

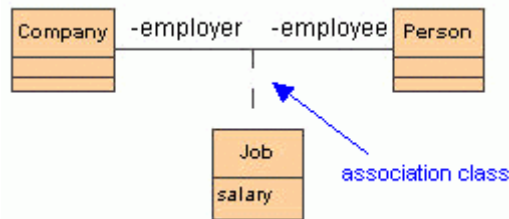
Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Options	Function
General Set a general information for the attribute.	Type	Shows an attribute type. It can be another project class or primitive type such as int or double or other. Select a type from drop-down combo box or click “...” button and select the type from the Select Element tree.
	Default Value	Type a value for an attribute or click “...” button and enter it in the Default Value dialog box.

Tab name	Options	Function
	Type Modifier	Additional information about the type. <ul style="list-style-type: none"> • & - one class has a reference to other model element. • * - one class has a pointer to other model element. • [] – one class has an array of other model elements.
	Navigable	Indicates whether it is possible to navigate across the property.
	Multiplicity	The multiplicity value of an attribute.
	Is Read Only	If <i>true</i> , the property may only be read, and not written.
	Is Static	This property scope means that the values returned by the property have no duplicates.
	Aggregation	When placed on a target end, specifies whether the target end is an aggregation with respect to the source end. Only one end can be an aggregation. <ul style="list-style-type: none"> • None. The end is not an aggregate. • Shared. The end is an aggregate; therefore, the other end is a part and must have the aggregation value of none. The part can be contained in other aggregates. • Composite. The end is a composite; therefore, the other end is a part and must have the aggregation value of none. The part is strongly owned by the composite and may not be part of any other composite.
	Is Derived	Specifies whether the property is derived, i.e., whether its value or values can be computed from other information.
	Is Ordered	If the multiplicity is greater than one, then the set of related elements can be ordered or unordered. <ul style="list-style-type: none"> • <i>False</i> - the elements form an unordered set. This is the default and need not be shown explicitly. • <i>True</i> - the elements of the set are ordered, but duplicates are still prohibited. This generic specification includes all ordering types.
	Is Unique	If <i>true</i> , the collection of values may not contain duplicates.

To create an association class

1. Draw two classes (for example A and B).
2. From the class diagram toolbar, expand the **Association** elements group and select the **Association Class** to draw.
3. Link the previously drawn classes with this path. An additional class with a dashed line will be created on the association between classes.



If an attribute type is another model class, this attribute can be represented as an association with a role (attribute name) between the owner class and the class of attribute type

Select **Create Roles** from the class shortcut menu. A list of available attributes opens. Select all or only one and an association relationship with a role is created.

NOTE

This command is visible only if one or more attribute types are other model classes.

To change an attribute name

The attribute name must be unique in the class scope.

1. Click the attribute in the selected class on the diagram pane or in the Browser tree.
2. Type a new name.
 - Change an attribute name in the **Attribute Specification** dialog box.

To define the type of an attribute

The attribute type can be of the other class, interface, or a primitive class, such as int or double.

- Select the type of an attribute from the **Type** drop-down list box in the **Attribute Specification** dialog box.

- Type a colon “:” and the name of the attribute type just after the attribute name on the diagram pane. If you specify a nonexistent type of an attribute, a new class is created.
- Click the “...” button in the **Type** field and select the type from the **Select Elements** tree.

To add additional information about the type of an attribute

1. Open the **Attribute Specification** dialog box.
2. Click the **Show Expert Properties** button to enlarge a list of general available attribute properties.
3. Select a sign in the **Type Modifier** drop-down list box:
 - & - one class has a reference to other model elements.
 - * - one class has a pointer to other model elements.
 - [] - one class has an array of other model elements.

To set the attribute visibility

Visibility Name	Function
Public ‘+’	an attribute can be accessed by any other elements.
Package ‘~’	an attribute can be accessed by elements from the same package.
Protected ‘#’	an attribute can be accessed from the inside of the selected class and classes derived from that class.
Private ‘-’	an attribute can be accessed only from inside of that class.

- Type ‘+’, ‘~’, ‘-’, or ‘#’ visibility marks just before an attribute name directly on a diagram.
1. Open the **Attribute Specification** dialog box.
 2. From the **Visibility** drop-down list box, select the desired item (public, package, protected, and private.).

NOTE

The attribute visibility is shown at the attribute signature.

To set an attribute scope

1. Open the **Attribute Specification** dialog box.

2. Click the **Show Expert Properties** button to enlarge the list of available attribute properties.
3. Select the **Is Static** check box.

To set the attribute multiplicity

1. Open the **Attribute Specification** dialog box (see above).
2. Click the **Show Expert Properties** button to enlarge the list of available attribute properties.
3. Select or set the multiplicity value in the **Multiplicity** drop-down box.

To set the attribute changeability

The attribute changeability controls the access by operations on the class on the opposite end.

Name	Function
Is Read Only	<p>When <i>false</i> - no restrictions on modifications.</p> <p>When <i>true</i> - the value may not be altered after the object is instantiated and its values initialized. No additional values can be added to a set.</p> <ol style="list-style-type: none">1. Open the Attribute Specification dialog box (see above).2. Click the Show Expert Properties button to enlarge the list of available attribute properties.3. Select/clear the Is Read Only check box.

Class

A class is drawn as a solid-outline rectangle with three compartments separated by horizontal lines. The top name compartment holds the class name and other general properties of the class (including stereotype); the middle list compartment holds a list of properties; the bottom list compartment holds a list of operations. The property and operation compartments are optional and you may suppress them.

A class is the descriptor for a set of objects with similar structure, behavior, and relationships. The model is concerned with describing the intention of the class, that is, the rules that define it. The run-time execution provides its extension, that is, its instances.

Classes are declared in the class diagrams and used in most of other diagrams. UML provides a graphical notation for declaring and using these classes as well as a textual notation for referencing classes within the descriptions of other model elements.

A class represents a concept within the system being modeled. It has a data structure, behavior, and relationships to other elements. The name of a class has a scope within the package in which it is declared and the name must be unique (among class names) within its package.

Working with classes

A general information about working with shapes is offered in Chapter 5, "Working With Diagrams."

All options associated with a class can be set in the **Class Specification** dialog box.

Refer to the "Specification dialog boxes" on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set general information about the class	Base Classifiers	Click "... " and assign an existing class from a model in the Select Elements dialog box or create a new one.
	Realized Interfaces	Click "... " and assign an existing interface from a model in the Select Elements dialog box or create a new one.
Ports	Name	Name of the class port.
	Type	Type, assigned to the port.
	Provided	Provided classifier is displayed.
	Required	Required classifier is displayed.
	Classifier	Name of classifier, owning port.
	Create	Creates a new port.
	Delete	Removes an existing port from the class.

Tab name	Box	Function
Signal Receptions		Manage the receptions of a class. For more information about receptions, see “Reception” on page 997.
Behaviors	Name	Name of the behavior.
	Type	Type, assigned to the behavior.
	Create	Choose item from the list - activity, interaction, state machine.
	Delete	Removes a behavior from the class.
Inner Elements Add another element to a .	Name	Model element name.
	Type	Model element type.
	Create	The corresponding specification dialog box opens. Define the selected model element in the dialog box.
	Delete	Removes the selected model element from the class.

To insert an inner element in the selected class

1. Double-click the selected class or select **Specification** from the class shortcut menu. The **Class Specification** dialog box opens.
2. Click the **Inner Elements** tab and then click the **Create** button. Select the element you wish to add from the list.
3. Click the selected element.
4. The corresponding **Specification** dialog box opens. Define the class, use case or interface, and click **OK**.

To show an inner element on the diagram

1. Select a created inner element in the Browser.
2. From the element shortcut menu, select **Create Symbol**.

To generate operations for setting or getting private data to the selected class

From the class shortcut menu, select **Tools** and then **Create Setters/Getters**. For a detailed description, see “Creating Setters / Getters” on page 475.

To control a list of operations and attributes that are visible on a diagram

Select **Edit Compartment** from the class shortcut menu. The **Compartment Edit** dialog box opens.

A class can be defined as active (a border to the class shape is added). An active class specifies whether an object of the class maintains its own thread of control.

A class is a generalizable element and can be defined as Abstract.

To define a class as abstract and/or active

1. Double-click the selected class or select **Specification** from the class shortcut menu. The **Class Specification** dialog box opens.
2. Select the **Is Abstract**, and/or **Is Active** check box in the **General** tab.

To show members (attributes and operations) on the classifier shape according to the visibility

From the shape shortcut menu, point to **Presentation Options**, and then select one of the check boxes in the **Show Members** subcommand. Possible choices:

- All;
- Only Public;
- Not Private.

Creating A Structured Class

There is a way to create a piece of model with a single click: class, with a composite structure diagram inside it and hyperlink them. It may be useful for architects and system engineers.

The same applies to SysML Block and IBD.

To create a structured class:

1. Create a class diagram.

2. In the class diagram toolbar, right-click the **Class** button. A menu with available options opens. Select the **Structured Class** element.
3. Click on the diagram. The class element with a hyperlink to a composite structure diagram is created.
4. Select the class in the Browser to see its structure. To do this, choose **Select in Containment Tree** on the diagram from the class shortcut menu, .

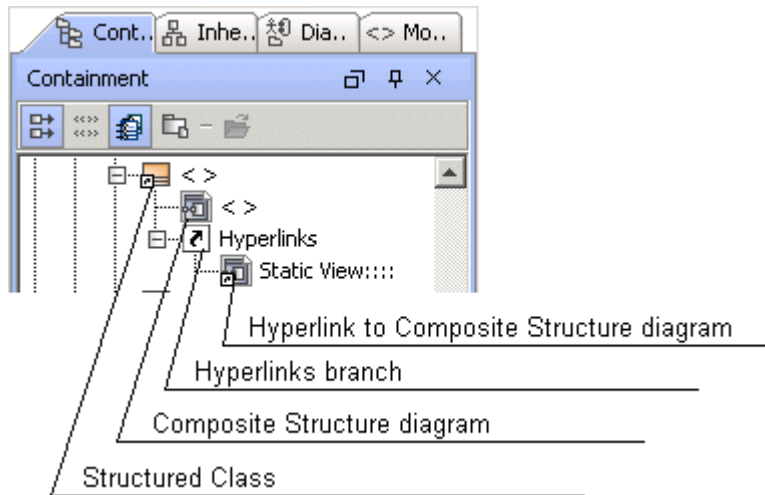


Figure 367 -- Example of structured class

See the above example of a structured class. In the Browser you may see the created class with the composite structure diagram inside it. The class is linked with the diagram. It means that after double click the diagram, or the Browser, the composite structure diagram will be opened.

For more information about hyperlink,s see "Defining Hyperlinks Between Elements" on page 362.

The names of the class and the composite structure diagram are synchronized. Type a name for the class, for example, Order, and the name of the diagram is automatically changed to Order and vice versa. This is synchronization of a diagram name and its context name.

For more information about diagram name and its context synchronization, see "Diagram name and its context name synchronization" on page 253.

Design Patterns

In MagicDraw, you may create and edit the design patterns for the selected class. A detailed description of templates can be found in the Design Patterns of Reusable Object-Oriented Software.

To create the design pattern for the selected class

1. From the class shortcut menu, select **Tools**, and then **Apply Pattern**. The **Pattern Wizard** dialog box opens.
2. Select the design pattern you want to apply and select the desired options. Click **OK**.
 - Select the class and then **Apply Pattern** from the **Tools** menu.

For a detailed description of this dialog box, see Section “Controlling Merge memory usage” on page -465.

Class presentation options

To organize a class data on the class shape

- Select **Presentation Options** from the class shortcut menu. The following choices are available in the **Presentation Options** submenu.
- The class presentation options can also be defined in the **Project Options** dialog box. For a detailed description of this dialog box, see Section “Project Options” on page -186.

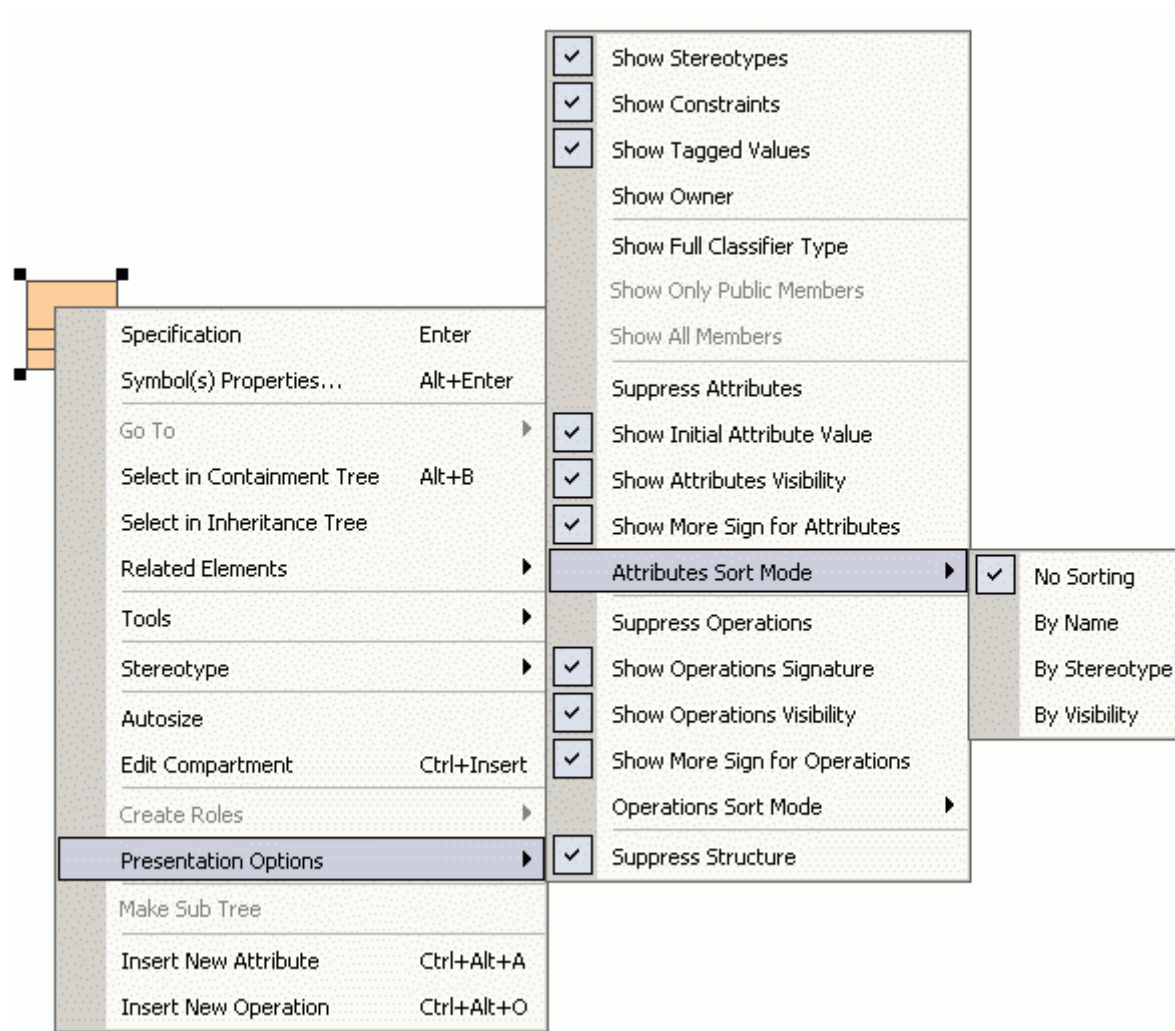


Figure 368 -- Presentation Options submenu

Command	Function (when selected)
Show Operations Signature	Shows the operation signature (arguments, return value, etc.).

Command	Function (when selected)
Show Full Classifier Type	Shows the full track of the type of an attribute from the root package.
Show Initial Attribute Value	Shows the initial attribute value.
Show Attributes Visibility	Shows the attribute visibility (public, package, private, or protected).
Show Operations Visibility	Shows the operation visibility (public, package, private, or protected).
Suppress Attributes	Attributes compartment is suppressed.
Suppress Operations	Operations compartment is suppressed.
Show Only Public Members	Shows only the public attributes and operations.
Show all Members	The default settings are restored and shows all attributes and operations.
Attributes Sort Mode <ul style="list-style-type: none"> • By Name • By Stereotype • By Visibility • No Sorting 	Choose the sorting parameter: <ul style="list-style-type: none"> • Sort attributes by name. • Sort attributes by stereotype • Sort attributes by visibility (public, package, private, or protected) • No sorting is executed.
Operations Sort Mode <ul style="list-style-type: none"> • By Name • By Stereotypes • By Visibility • No Sorting 	Choose the sorting parameter: <ul style="list-style-type: none"> • Sort operations by name. • Sort operations by stereotype. • Sort operations by visibility (public, package, private, or protected) • No sorting is executed.
Show More Sign For Attributes	Additional information sign '...' in the class attributes list, when a portion of attributes are omitted by editing a class compartment.
Show More Sign For Operations	Additional information sign '...' in the class operations list, when a portion of operations are omitted by editing a class compartment.
Show Stereotypes	Shows the stereotypes on a class.

Command	Function (when selected)
Show Constraints	Shows the constraints on a class.
Show Tagged Values	Shows the tagged values on a class.
Show Owner	Shows the owner's (package, subsystem, or model) name on a class.

Collaboration

A collaboration is represented as a kind of classifier. It defines a set of cooperating entities to be played by instances (its roles) as well as a set of connectors that define the communication paths between the participating instances. The cooperating entities are the properties of the collaboration.

A collaboration specifies a view (or projection) of a set of cooperating classifiers. It describes the required links between instances that play the roles of the collaboration as well as the required features of the classifiers that specify the participating instances. Several collaborations may describe different projections of the same set of classifiers.

Define a collaboration in the **Collaboration Specification** dialog box.

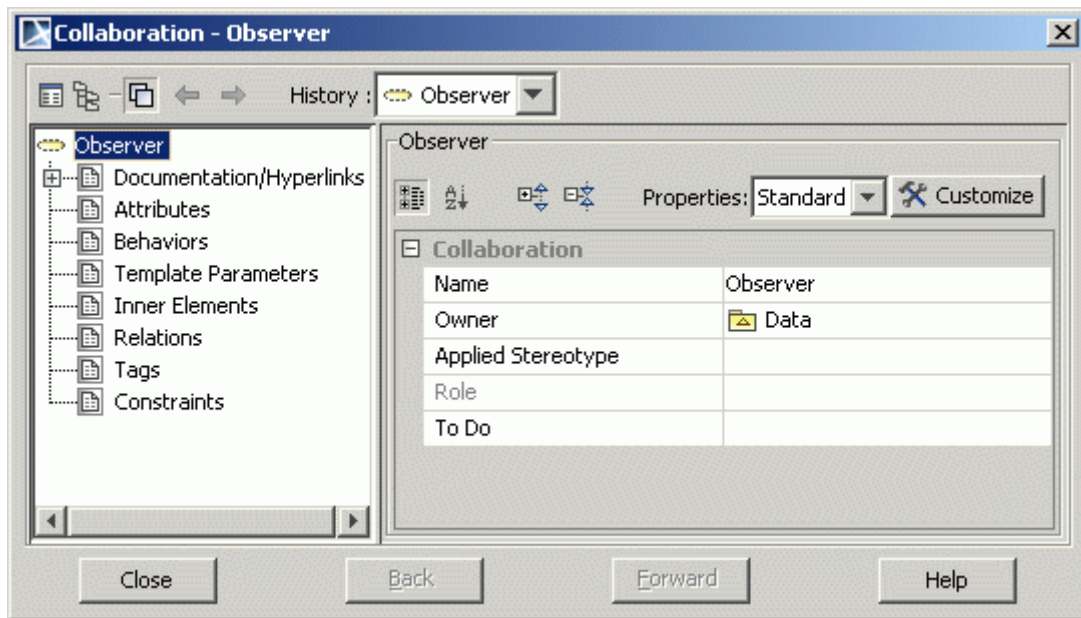


Figure 369 -- Collaboration Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for the information of the specification elements not covered in this section.

Tab name	Box	Function
Behaviors Any behavior attached to the collaboration type applies to the set of roles and connectors bound within a given collaboration use.	Name	Name of the added behavior.
	Type	Type of the behavior
	Create	Select the kind of behavior you want to create - Activity, Interaction, or State Machine. The corresponding Specification dialog box opens.
	Delete	Remove the selected behavior from the collaboration.

Tab name	Box	Function
Inner Elements	Name	Name of the inner collaboration element.
	Type	Type of the inner collaboration element.
	Create	Select an element from the list. The corresponding Specification dialog box opens. Define the element and click Back to return to Collaboration Specification .
	Delete	Remove the inner element from the collaboration.

To assign a new behavior to the collaboration

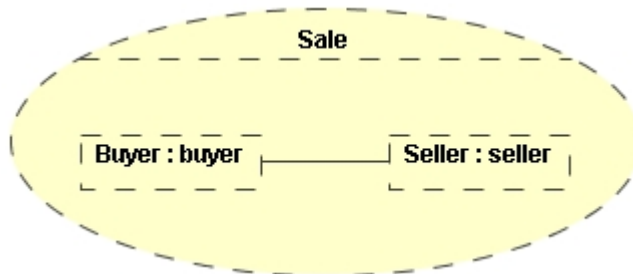
- In the **Collaboration Specification** dialog box, **Behaviors** tab, click the **Create** button. Define a behavior in the open **Specification** dialog box.
- From the collaboration shortcut menu in the Browser, select **New**, and then **Activity**, **Interaction**, or **State Machine**. Type the name of the behavior and modify it in the open **Specification** dialog box.

NOTE

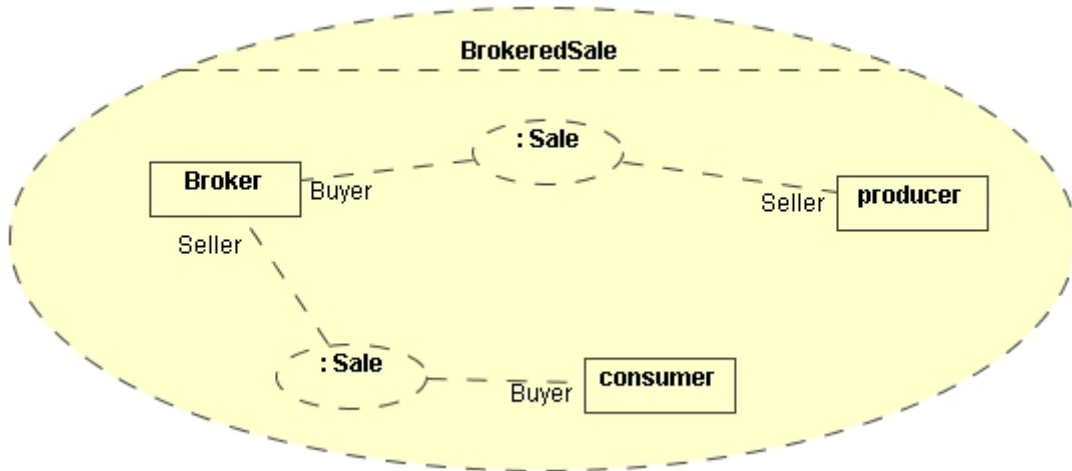
You can edit the assigned classifier role in the **Collaboration Specification** dialog box, **Behaviors** tab.

To draw the Collaboration

1. From the Composite Structure diagram toolbar, select the Collaboration to draw on the diagram pane and type the collaboration name.
2. Draw Part from the Composite Structure diagram toolbar and from the shortcut menu, **Type** submenu, select an already existing (or create a new) classifier as the part type.
3. From the Composite Structure diagram toolbar, select the Connector to connect the parts.



4. From the Composite Structure diagram toolbar, select the Collaboration Use element to draw on the collaboration. Select an already existing, or create new collaboration for the Collaboration Use.
5. From the Composite Structure diagram toolbar, select the Role Binding path to connect the parts and Collaboration Use role, which can be selected in the **Select Role** dialog box.



Combined Fragment

A fragment is an abstract notion of the most general interaction unit. It is a piece of an interaction. Each interaction fragment is conceptually like an interaction by itself. Using the Combined Fragment, a fragment of a sequence diagram can be separated.

MagicDraw represents twelve kinds of fragments: Alternatives, Loop, Option, Parallel, Break, Negative, Critical Region, Consider, Ignore, Weak Sequencing, Strict Sequencing, and Assertion.

Tab name	Property	Function
General	Interaction Operator	Select an operator for the fragment. Available options: seq, alt, opt, break, par, strict, loop, critical, neg, assert, ignore, consider.

Tab name	Property	Function
Formal Gates Actual Gates		In the Formal Gates and Actual Gates panes are listed inside or outside combined fragment created gates. For more information about gates, see “Gate” on page 894.
Operands	Guard	Separated alternative parts of the interaction fragment.
	Create	Creates a new guard value.
	Delete	Removes a value from the list.
	Up	Moves an item to an upper position in the list.
	Down	Moves an item to a lower position in the list.

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

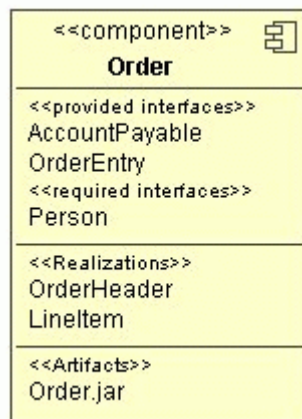
Component

A component represents all kinds of elements pertaining to piecing together software applications. They can be simple files, such as DLLs or executables.

According to UML it is possible to list component properties. The following compartments are available for the component:

- Provided/required interfaces
- Realizations

- Artifacts



For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the selected component in the **Component Specification** dialog box.

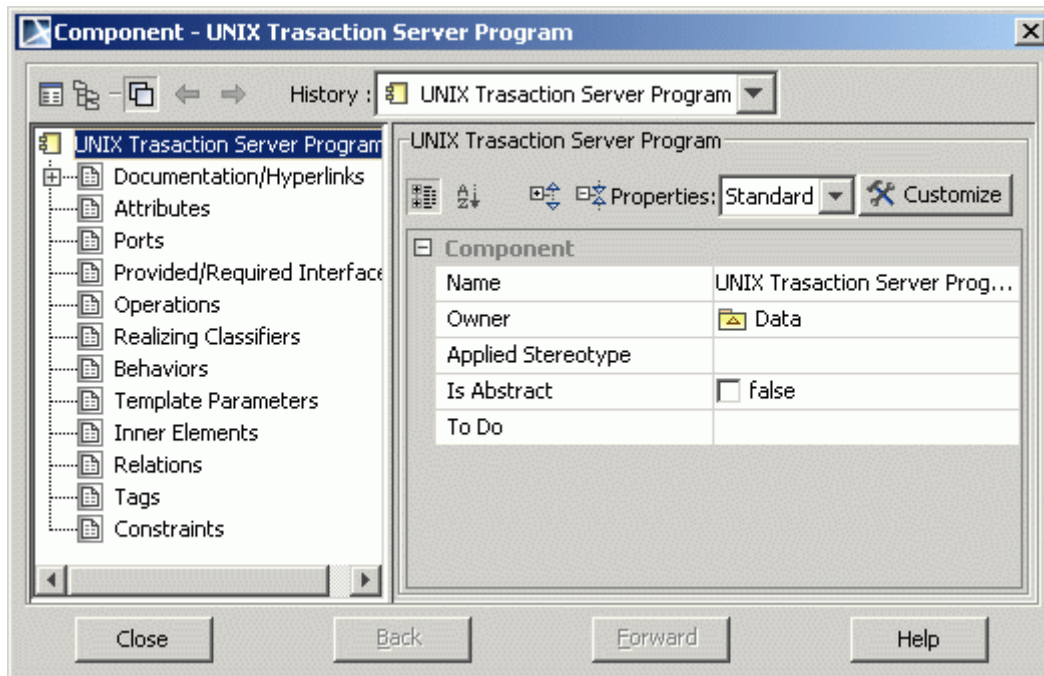



Figure 370 -- Component Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
Ports	Name	Name of the port.
	Type	Type, assigned to the port.
	Provided	The provided classifier is displayed.
	Required	The required classifier is displayed.
	Classifier	Name of the classifier, owning a port.
	Create	Creates a new port.
	Delete	Removes an existing port from the component.

Tab name	Box	Function
Provided/Required Interfaces	Name	Name of the provided/required interface.
	Type	Possible values - provided or required.
	Add	Adds the provided/required interface.
	Remove	Removes the provided/required interface from the list.
Realizing Classifiers	Name	Name of the realizing classifier
	Type	Type of the classifier (class, interface, etc.)
	Owner	The model element name, owning a classifier.
		Click  button to open the classifier Specification dialog box.
	Add	The Select Elements dialog box opens. Select an element from the model element tree.
	Remove	Removes a realizing classifier from the list.
Behaviors	Name	Name of the behavior.
	Type	Type, assigned to the behavior.
	Create	Select an item from the list - activity, interaction, state machine.
	Delete	Removes a behavior from the class.
Inner Elements Add another element to a component.	Name	The model element name.
	Type	The model element type.
	Create	Select an element from the list. The corresponding specification dialog box opens. Define the selected model element in the dialog box.
	Delete	Removes the selected model element from the component.

Displaying interfaces, realizations, and artifacts on the component

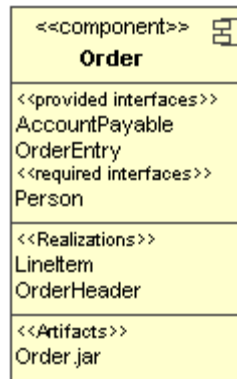
The following component shape compartments can be displayed or hidden:

- Provided/Required interfaces

- Realizing classifiers
- Manifested artifacts.

To show/hide the interfaces, realizations, and artifacts on the component shape:

- From the component shape shortcut menu, select **Presentation Options** and then clear/select **Suppress Interfaces**, **Suppress Realizations**, or **Suppress Artifacts** check box.
- From the component shape shortcut menu, select **Symbol(s) Properties**. Change the same check box values in the open **Component Properties** dialog box.



Connector

Specifies a link that enables communication between two or more instances. This link may be an instance of an association, or it may represent the possibility of the instances being able to communicate because their identities are known by virtue of being passed on as parameters, held in variables or slots, or because the communicating instances are the same instance.

The link may be realized by something as simple as a pointer or by something as complex as a network connection. In contrast to the associations, which specify the links between any instance of the associated classifiers, the connectors specify the links between instances playing the connected parts only.

Each connector may be attached to two or more connectable elements, each representing a set of instances.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the selected connector in the **Connector Specification** dialog box.

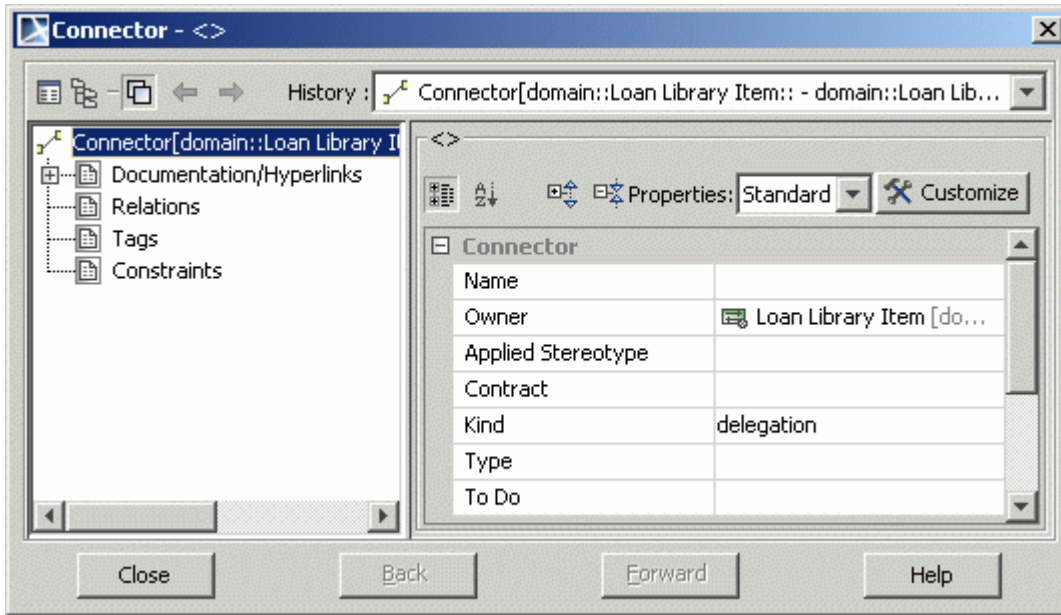


Figure 371 -- the Connector Specification dialog box

Refer to the "Specification dialog boxes" on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set a general information about the include relationship.	Owner	The name of the element, which owns a connector.
	Contract	Assign an activity, interaction, or state machine to a connector. Click the '...' button and select the elements in the Select Elements dialog box.
	Kind	Possible choices: <ul style="list-style-type: none"> • delegation (default) • assembly

To open the **Connector End Specification** dialog box

1. In the Browser tree, expand Interaction, **Relations** branch and in the Connector group, select one of the roles. Select **Specification** from the shortcut menu, or double-click the role. The **Connector End Specification** dialog box opens.

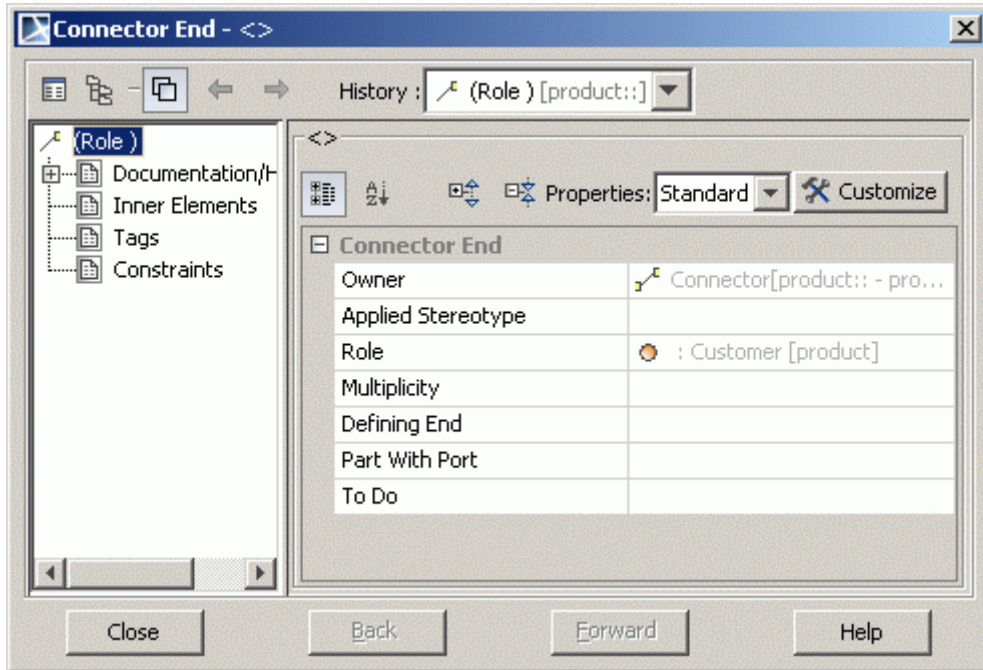


Figure 372 -- the Connector End Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General Set general information about the link end	Owner	The name of the connector, which owns the connector end.
	Role	The name of the property, representing a lifeline.
	Multiplicity	The multiplicity value of the connector end.

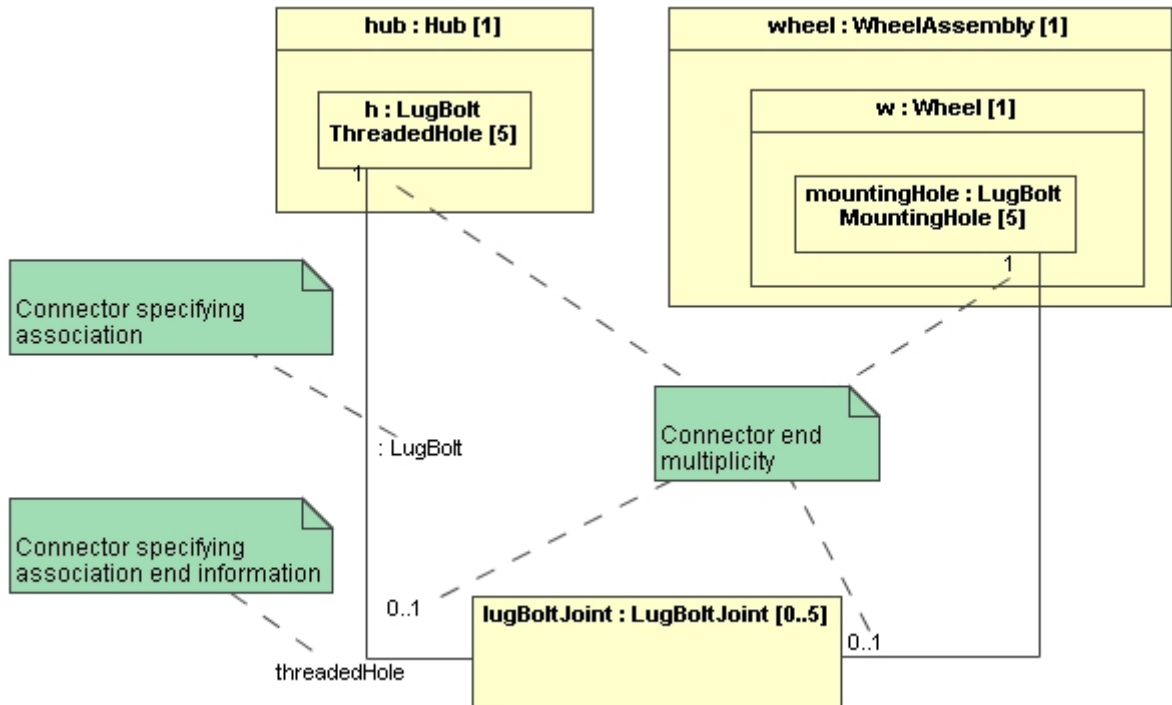
11 MODEL ELEMENTS

Connector

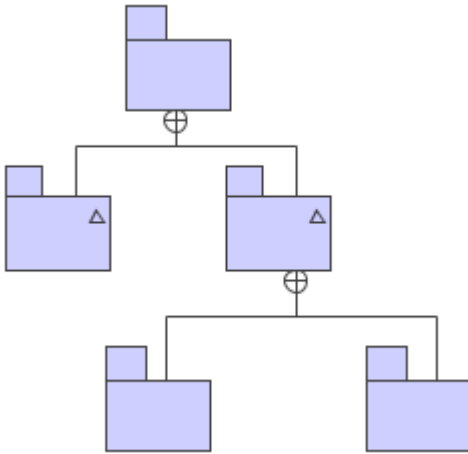
Tab name	Box name	Function
	Defining End	When a connector has an association specifying the type of the connector, connector end, this property refers to the appropriate association end.
	Part With Port	Refers to the part containing the port that the connector is attached to.

Drawing a connector end

The connector end information can be displayed on the diagram.



Containment



A containment shows a class, package or other model element declared within another model element. Such a declared class is not a structural part of the enclosing class but merely has scope within the namespace of the enclosing class, which acts like a package toward the inner class.

Data type

A data type is a type whose values have no identity; that is, they are pure values. It is a classifier and inherits the general features of the classifier: visibility, generalizable element properties, and operations.

MagicDraw provides the following predefined data types: boolean, byte, char, date, double, float, int, Integer, long, short, void, and String.

You may also create Enumeration or Primitive data types.

To create a new Data Type, including Enumeration or Primitive

- From the Browser, select **New** from the class or **New Element** from the package, subsystem, or model shortcut menu, and then select **Data Type**, **Enumeration**, or **Primitive**.
- In the **Class**, **Package**, **Subsystem**, and **Model Specification** dialog boxes, **Inner Elements** tab, click **Create** and select a data type.

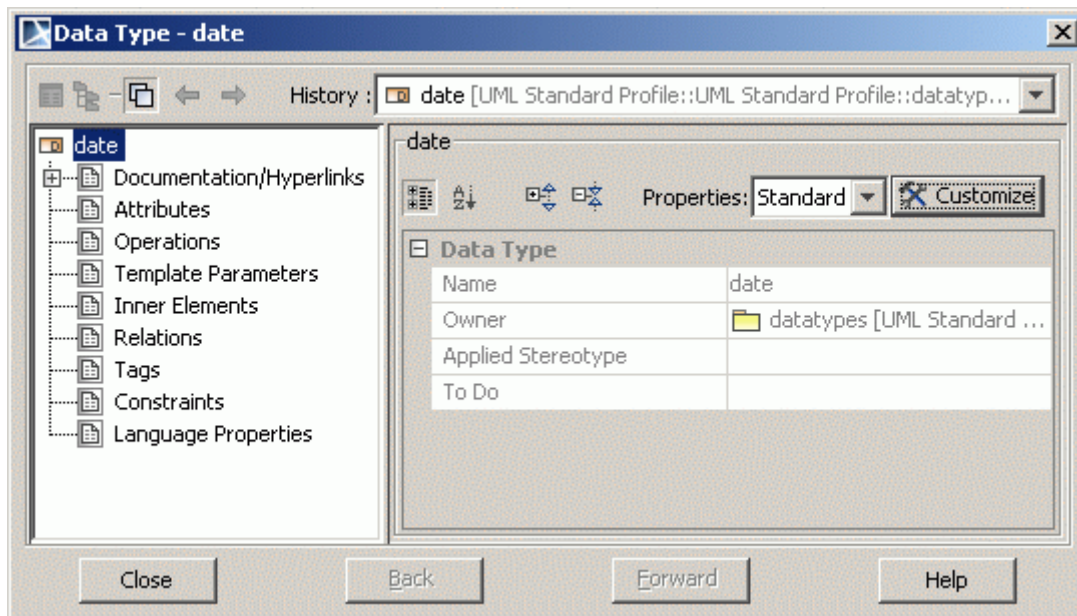
NOTE:

You may create an enumeration by clicking the **Enumeration** button from the class diagram toolbar:



To create a symbol of the created data type

From the created data type shortcut menu in the Browser, select the **Create Symbol** command.



Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set general information about the data type Inner Elements Add constraint to data type.	Owner	Shows an element, which contains the data package. The default data packages contain the <i>UML Standard Profile</i> .
	Name	Displays a constraint name.
	Type	Shows an item type, in this case Constraint. Click the button to open the Constraint Specification dialog box.
	Create	The Constraint Specification dialog box opens. Define the constraint.
	Delete	Removes the selected constraint from the inner elements list.

Enumeration

The enumeration defines a kind of data type whose range is a list of predefined values, called enumeration literals. An Enumeration may contain operations, but they must be pure functions (this is the rule for all data type elements).

Define an enumeration in the **Enumeration Specification** dialog box.

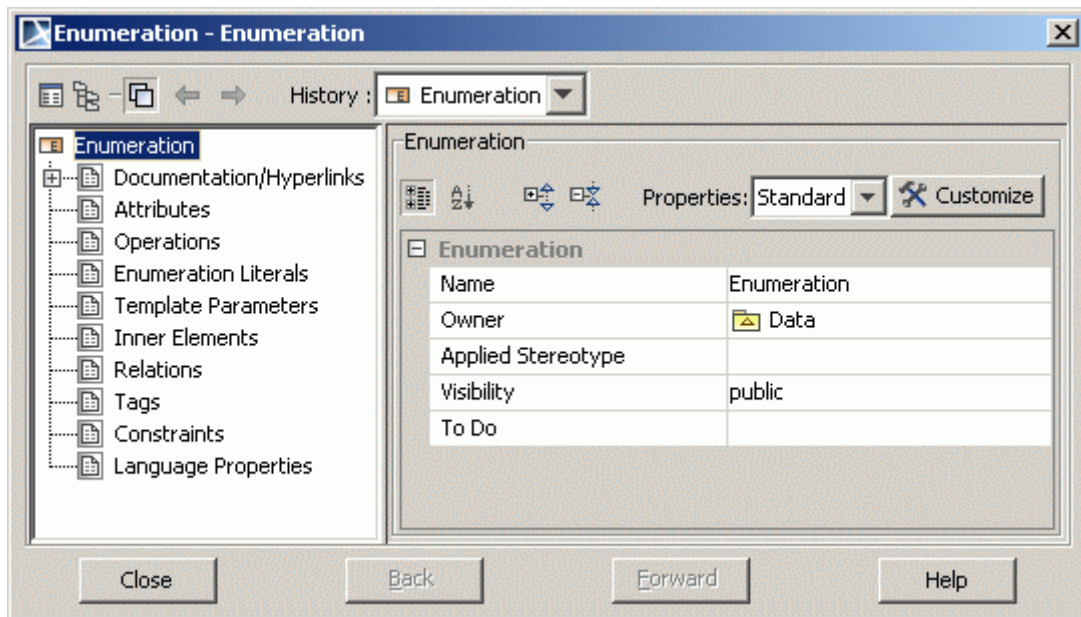


Figure 373 -- Enumeration Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set a general information about the enumeration	Owner	Shows the name of the element, which contains an enumeration.
Enumeration Literals An enumeration literal defines an element of the run-time extension of an enumeration data type.	Name	The name of the enumeration literal.
	Enumeration	The owner of the enumeration literal. Click the button to open the Enumeration Literal Specification dialog box and edit the information about the enumeration literal.
	Create	The Enumeration Literal Specification dialog box opens. Define a new enumeration literal.
	Delete	Removes the selected enumeration literal from the enumeration.

Tab name	Box	Function
Inner Elements Add a constraint to an enumeration	Name	Displays a constraint name.
	Type	Shows an item type, in this case Constraint. Click the button to open the Constraint Specification dialog box.
	Create	The Constraint Specification dialog box opens. Define the constraint.
	Delete	Removes the selected constraint from the inner elements list.

To add an enumeration literal

An enumeration literal defines an element of the run-time extension of an Enumeration data type. It has no relevant substructure, therefore, it is atomic.

1. Open the **Enumeration Specification** dialog box.
2. In the **Enumeration Literals** tab, click **Create** button.
3. The **Enumeration Literal Specification** dialog box opens. Define an enumeration literal. Click **Back** to return to **Enumeration Specification** dialog box.
 - Choose the **Insert New Enumeration Literal** command from the **Enumeration** shortcut menu.

To suppress the enumeration literals

1. From the enumeration shortcut menu, select **Presentation Options**.
2. Select the **Suppress Enumeration Literals** check box.
 - Use the smart manipulation button with minus sign on the diagram pane, enumeration symbol.

To open the **Enumeration Literal Specification** dialog box

1. Open the **Enumeration Specification** dialog box.
2. In the **Enumeration Literals** tab expand tree, double-click the desired literal, or click the **Edit**, or **Create** button.

Primitive

A primitive defines a predefined data type without possessing any relevant UML substructure; that is, it has no UML parts. A primitive data type may have an algebra as well as operations defined outside of UML (for example, mathematically). The primitive data types used in UML include Integer, Unlimited Integer, and String.

Decision Node

Decisions are made using guard conditions. They help protect transitions that depend on a guarding condition. The symbol used for the decision is a large diamond shape, which may have one or more incoming transitions and two or more outgoing transitions.

A decision in an activity diagram is used much like a choice or junction point in the state diagrams. Decision points allow to separate and merge the transition paths back together.

The Decision Node can be converted to a Merge Node. Right-click to open the shortcut menu and select Convert to Merge Node.

Dependency

A dependency is a relationship signifying that a single or a set of model elements require

other model elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s).

A dependency is shown as a dashed arrow between classes or packages. The model element at the tail of the arrow (the client element) depends on the model element at the arrowhead (the supplier element). The arrow can be labeled with an optional stereotype and an optional individual name.

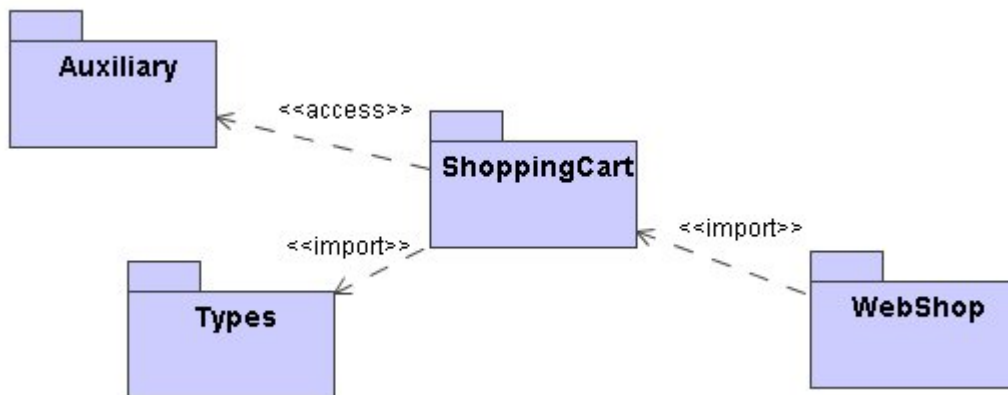
NOTE

You may also draw a dependency between a class and other class elements, such as attributes and operations.

For more information about working with the symbols, see Chapter 5, "Working With Diagrams."

Example of the dependency relationships

See the example of dependency relationships in the Figure below.



The Dependency and Its Kinds Specification dialog boxes

Dependency, abstraction, and usage relationships defined in the dialog box of the same structure. They differ from one another only by the corresponding Specification name.

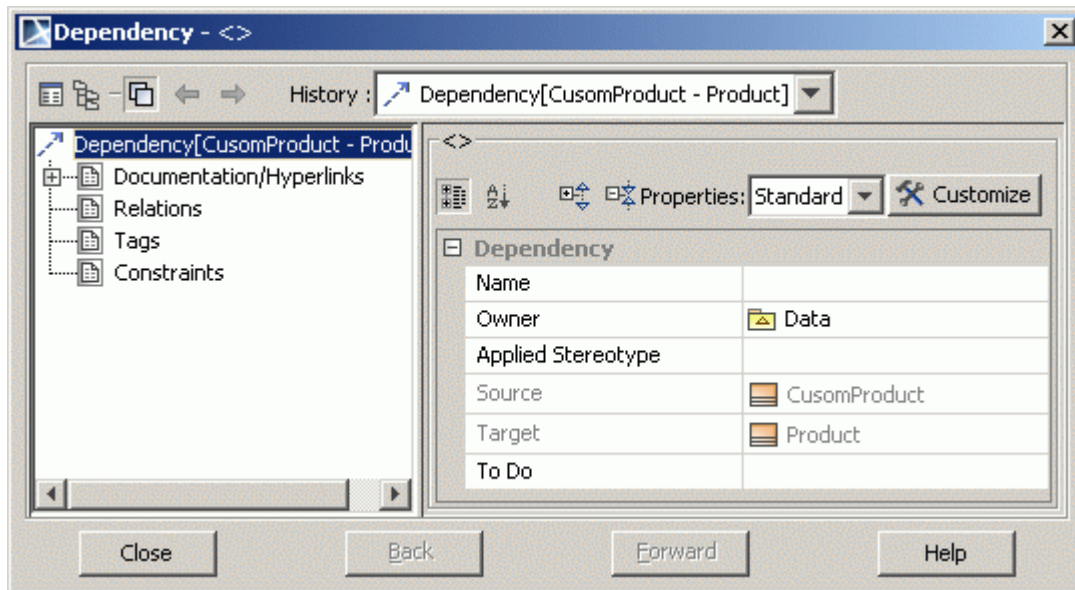


Figure 374 -- Dependency Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General	Source	Shows the name of the dependency's supplier element.
	Target	Shows the name of the dependency's client element.

Template Binding dependency

Term The OMG UML specification (UML 2.2: Superstructure) states:

“A template binding represents a relationship between a templateable element and a template. A template binding specifies the substitutions of actual parameters for the formal parameters of the template.”

Define a binding dependency in the **Template Binding Specification** dialog box.

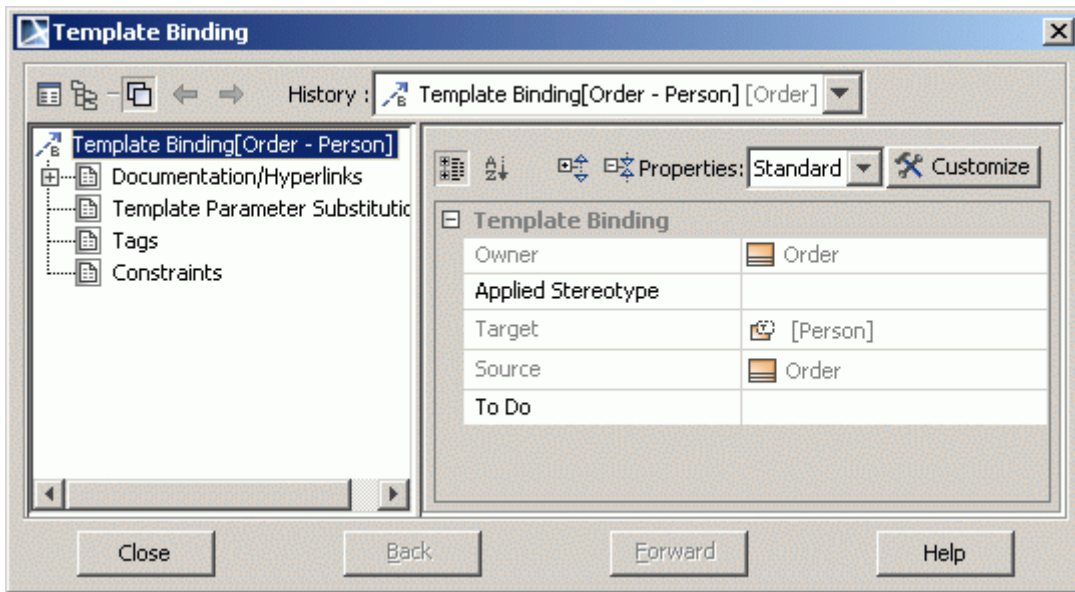


Figure 375 -- Binding Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General	Source	Shows the name of the dependency's supplier element.
	Target	Shows the name of the dependency's client element.

Tab name	Box name	Function
Template Parameter Substitutions The parameter substitutions owned by this template binding.	Name	The name of the template parameter.
	Type	The type of the template parameter.
	Actual	A reference to a formal template parameter of the target template signature.
	Create	The Select Template Parameter dialog box opens. Select the item from the list and click OK .
	Delete	Removes the template parameter substitution.

Abstraction

An abstraction is a relationship that relates two elements or sets of elements that represent the same concept at different levels of abstraction or from different viewpoints. In the metamodel, an abstraction is a dependency in which there is a mapping between the supplier and the client.

Define an abstraction relationship in the **Abstraction Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 881.

Usage

A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation. In the metamodel, a usage is a dependency in which the client requires the presence of the supplier.

Define a usage relationship in the **Usage Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 881.

Package Merge

A package merge is a directed relationship between two packages, that indicates that the contents of the two packages are to be combined. It has a dependency relation with the applied stereotype `<<merge>>`.

Define a merge relationship in the **Dependency Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 881.

Package Import

A package import is defined as a directed relationship that identifies a package whose members are to be imported by a namespace. It is a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace. It is dependency relation with applied stereotype `<<import>>`.

Define an import relationship in the **Dependency Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 881.

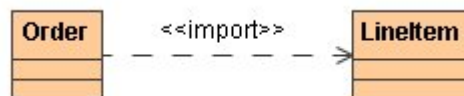
To draw the Package Import link, select the Package Import path to draw in the Class diagram toolbar, from the Abstraction group.

Element Import

An element import is defined as a directed relationship between an importing namespace and a packageable element. The name of the packageable element or its alias is to be added to the namespace of the importing namespace. It has a dependency relation with the applied stereotype `<<import>>`.

Define an import relationship in the **Dependency Specification** dialog box. For a detailed description of this dialog box, see “The Dependency and Its Kinds Specification dialog boxes” on page 881.

To draw the Element Import link, select the Element Import path to draw in the Class diagram toolbar, from the Abstraction group, .



Access

An access relationship shows that elements can only be accessed from a package, and it cannot be referenced.

To draw an Access link:

3. In the Class diagram toolbar, from the Abstraction group, select the Package Import path to draw.
4. Open the **Package Import Specification** dialog box and set the **Visibility** property to *private*.

Deployment

To draw a deployment link

1. In the implementation diagram toolbar, click the Deployment button and draw a deployment link from a node to an artifact.
2. From the node shortcut menu, Presentation Options submenu, clear the Suppress Deployment check box to display the deployed artifacts on the node instance shape.

Deployment Specification

The Deployment Specification is a type of Artifact.

To draw the Deployment Specification on the diagram pane

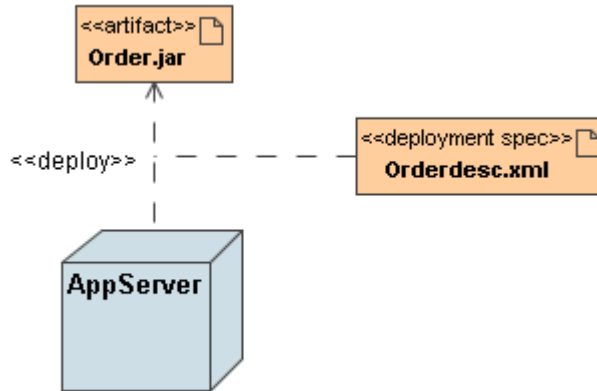
In the Implementation diagram toolbar, right-click the Artifact button group. In the open list, select the Deployment Specification to draw.

The Deployment Specification is a general mechanism to parameterize a Deployment relationship.

To specify the Deployment relationship

1. Create the Deployment Specification.

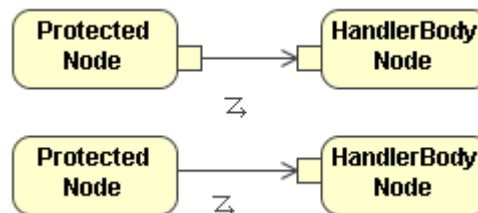
2. Between the Deployment relationship and Deployment Specification draw the Dependency relationship. The Dependency is drawn without an arrow.



Exception Handler

An Exception Handler is an element that specifies a body to execute in case the specified exception occurs during the execution of the protected node.

The Exception Handler may be drawn from an Output Pin to an Input Pin or from an Action to an Input Pin:



Extend

A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case. The extension takes place at one or more specific extension points defined in the extended use case.

Define the extend relationships in the **Extend Specification** dialog box.

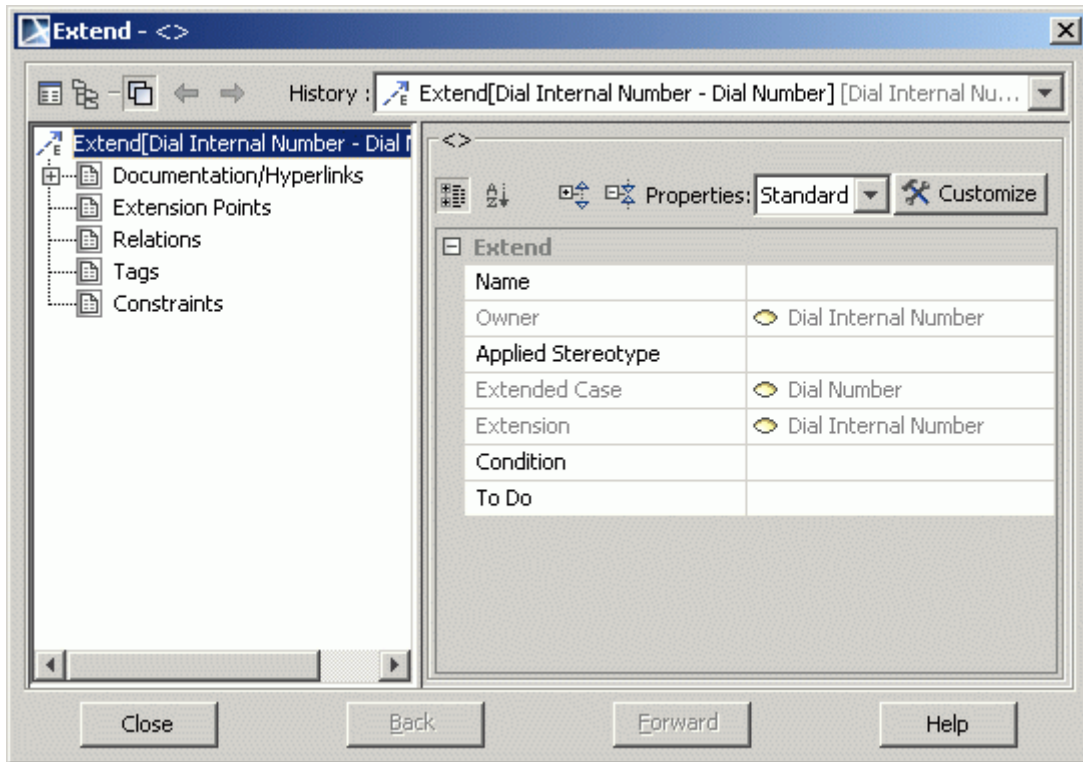


Figure 376 -- the Extend Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Group name	Box	Function
General Set general information about the extend relationship	Condition	An expression specifying the condition, which must be fulfilled if the extension is to take place.
Extension Points Select the extension points to be assigned to the extend relationship	Assign	Select predefined extension points you wish to assign to the extend relationship in the Extension Points dialog box.
	Unassign	Remove an already created extension point from the use case.
	Up	Move the selected extension point to an upper position.
	Down	Move the selected extension point to a lower position.

To open the **Extension Point Specification** dialog box

1. Select **Specification** from the use case shortcut menu, or double-click the use case shape. The **Use Case Specification** dialog box opens.

- Expand the **Extension Points** group and then click on the created extension point.
(Double-clicking opens an independent **Extension Points Specification** window).

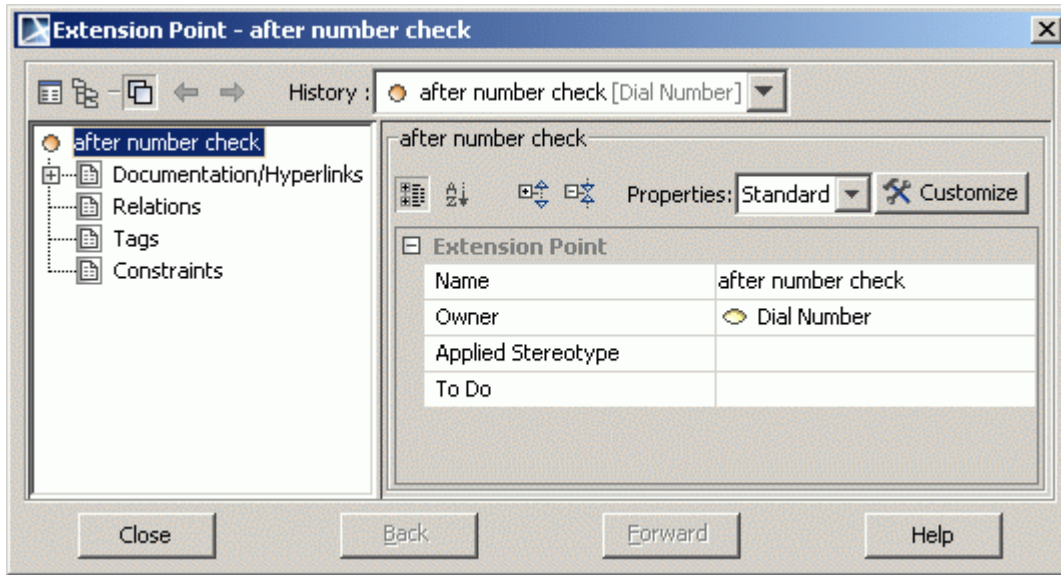


Figure 377 -- Extension Point Specification dialog box

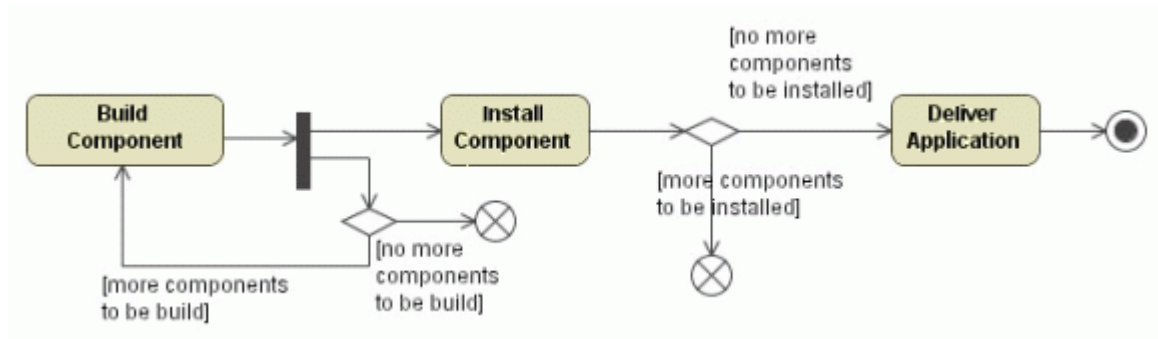
Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General A general information about the extension point is displayed.	Owner	The name of the use case, which owns the extension point.

Flow Final Node

It is a final node that terminates a flow and destroys all tokens that arrive at it. It has no effect on other flows in the activity.

See the following example for the flow final element notation in a MagicDraw project.



For more information about defining the flow final node, see “Containment” on page 874.

Fragment

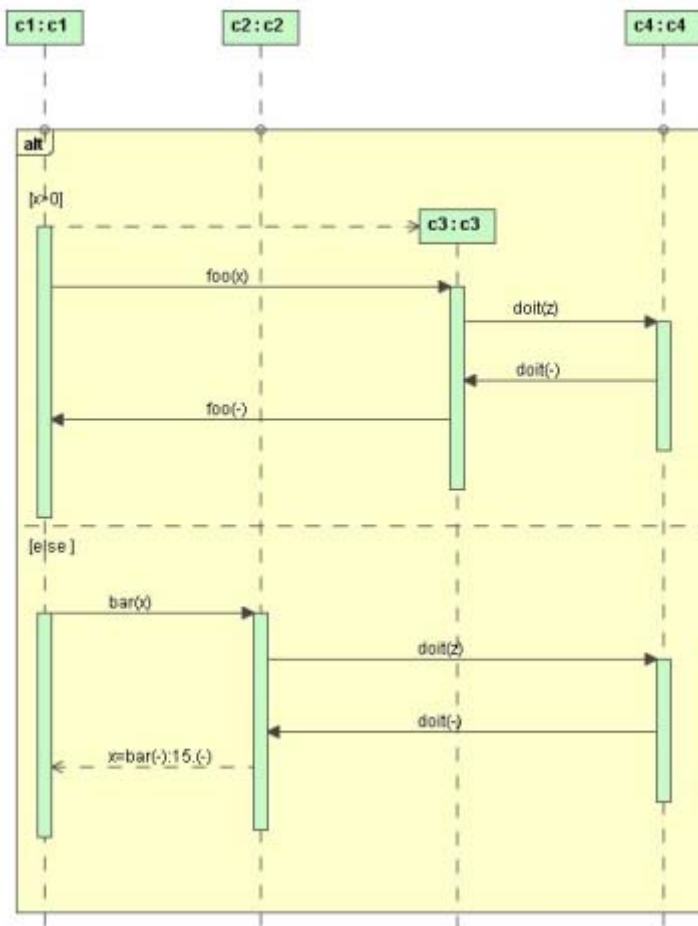
MagicDraw represents twelve kinds of fragments: Alternatives, Loop, Option, Parallel, Break, Negative, Critical Region, Consider, Ignore, Weak Sequencing, Strict Sequencing, and Assertion.

Alternative Fragment

The alternative fragment models *if...then...else* constructions.

To draw an Alternative Fragment

- In the Sequence diagram toolbar, select the Alternatives element to draw.



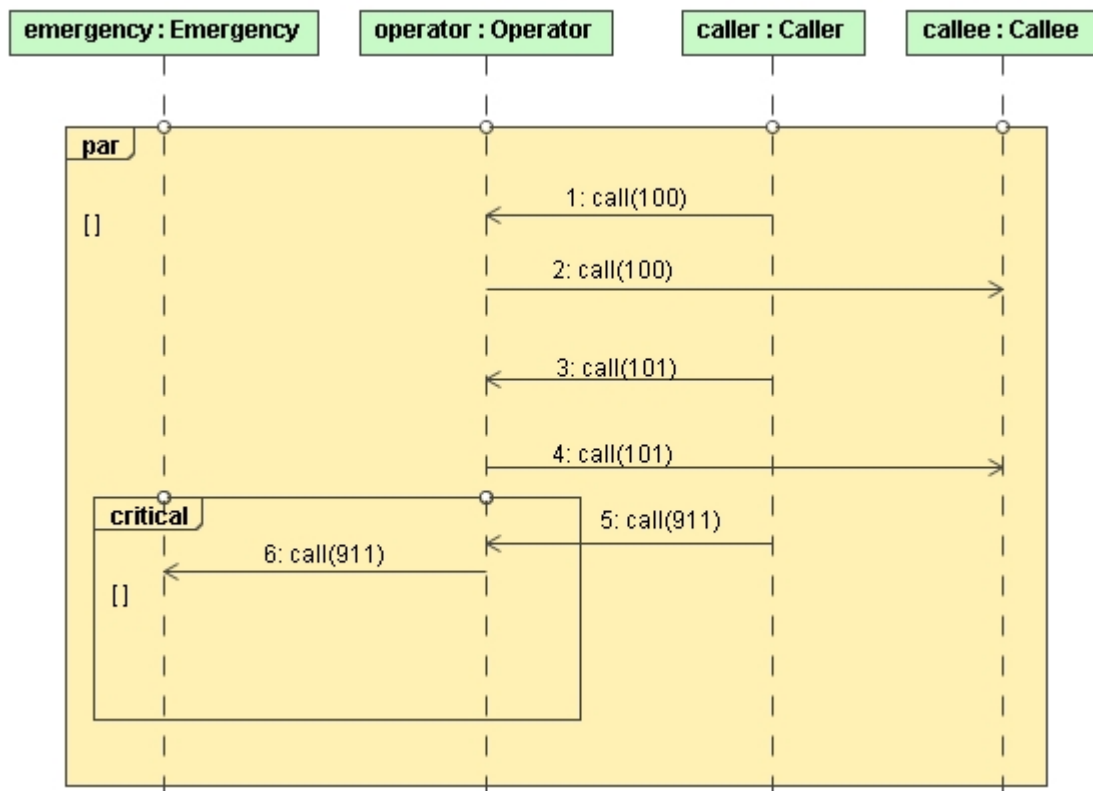
Combined Fragment

The UML Combined Fragment element allows the expressions of interaction fragments to be defined in the sequence diagram. The combined fragments provide a means to define special conditions and subprocesses for any sections of lifelines in the sequence diagram by specifying an area where the

conditions or subprocesses apply. Using the Combined Fragment, a fragment of the sequence diagram can be separated.

To draw a Combined Fragment

- In the Sequence diagram toolbar, select the Alternatives element to draw.
- From the combined fragment shortcut menu select the **Covered Lifelines** command and in the **Covered Lifelines** dialog box, select the lifelines to display.
- In the Sequence diagram toolbar, **Options** group, select the Parallel combined fragment to draw.
- Draw the Critical Region combined fragment from the Sequence diagram diagram toolbar, **Option** button group.




Function Behavior

Term	<p>The OMG UML specification (UML 2.2: Superstructure) states:</p> <p>“A function behavior is an opaque behavior that does not access or modify any objects or other external data.”</p> <p>“Primitive functions transform a set of input values to a set of output values by invoking a function. They represent functions from a set of input values to a set of output values. The execution of a primitive function depends only on the input values and has no other effect than to compute output values. A primitive function does not read or write structural feature or link values, nor otherwise interact with object memory or other objects. Its behavior is completely self-contained. Specific primitive functions are not defined in the UML, but would be defined in domain-specific extensions. Typical primitive functions would include arithmetic, Boolean, and string functions.</p> <p>During the execution of the function, no communication or interaction with the rest of the system is possible. The amount of time to compute the results is undefined. FunctionBehavior may raise exceptions for certain input values, in which case the computation is abandoned.”</p>
-------------	--

The function behavior allows modeling external functions that take only inputs and produce outputs. It has no effect on the specified system.

To create the function behavior

In the Browser tree, right-click the Data package. In the shortcut menu, select the **New Element** command and then select Function Behavior .

Parent topic: “Model Elements” on page 796.

Gate

Term	<p>The OMG UML specification (UML 2.2: Superstructure) states:</p> <p>“A Gate is a connection point for relating a Message outside an Interaction-Fragment with a Message inside the InteractionFragment.”</p> <p>“Gates are connected through Messages. A Gate is actually a representative of an OccurrenceSpecification that is not in the same scope as the Gate. Gates play different roles: we have formal gates on Interactions, actual gates on InteractionUses, expression gates on CombinedFragments.”</p>
-------------	---

MagicDraw version 15.0 and later allows the display of messages leaving or entering a sequence diagram, interaction use, or combined fragment. The Gate is a connection point for representing a message from the outside to the current sequence diagram, interaction use, or combined fragment.

Gates can be used in three cases:

- For passing and returning arguments to InteractionUse, which calls some Interaction.
- For displaying “exceptions” as messages that stops an interaction execution and leaves it.
- For “calling” sequence blocks represented as CombinedFragments.

Gate has no notation. Gates are created as message ends when drawing messages to/from a diagram frame, an interaction use, or a combined fragment.

Parent topic: “Model Elements” on page 796.

Usage in diagrams: “Sequence Diagram” on page 665.

Related topics:

“Model” on page 938.

“Lifeline in the Sequence Diagram” on page 925.

“Combined Fragment” on page 865.

“Interaction Use” on page 916.

To create a formal gate

- Draw a call, send, create, or delete a message from the diagram frame.
- Draw a reply message to the diagram frame.
- Draw a call, send, create, or delete a message from the combined fragment (inside combined fragment).

NOTES

- You can view the created gates of message in the **Message** specification dialog box. In the **Send Event** list box you may see formal gate and the **Receive Event** lists the actual gate.
- The Gate uses text from the message as an identification name. For example, a message name or a message operation.

To draw create an actual gate of the formal gate

- Draw a call, send, create, or delete a message to the interaction use, which refers to the diagram with the formal gates. The **Select Formal Gate** dialog box opens.
- Draw a reply message from the interaction use, which refers to the diagram with the formal gates. The **Select Formal Gate** dialog box opens.
- Draw a call, send, create, or delete a message to the combined fragment (outside combined fragment). The **Select Formal Gate** dialog box appears.

NOTE

You can also view the formal and actual gates in the gates **Interaction** specification dialog box, **Interaction Use** dialog box, **Combined Fragment** dialog box, and the **Actual Gates** and **Formal Gates** panes.

To select a formal gate for the actual gate

1. Draw a message to invoke the **Select Formal Gate** dialog box (see “To draw create an actual gate of the formal gate” on page 895).
2. Select one of the listed formal gates and click OK. An actual gate is created.
 - Or from the message shortcut menu, select the **Select Formal Gate** command. The **Select Formal Gate** dialog box opens.

NOTE

The **Select Formal Gates** command exists only if there are formal gates.

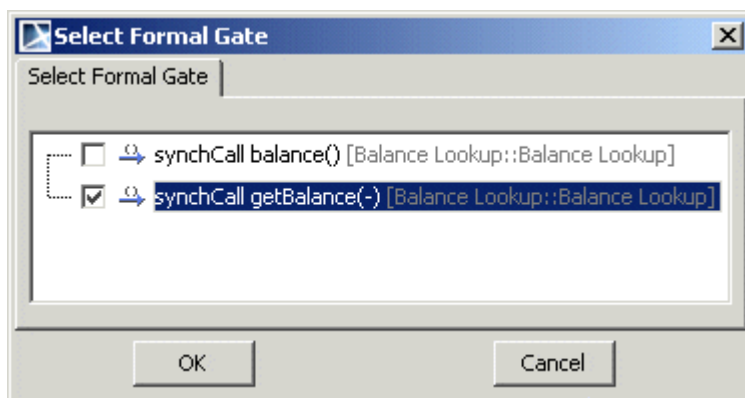


Figure 378 -- The Select Formal Gate dialog box

The formal gate and actual gate usage in the sequence diagram

See Figure 379 on page 897 where the *getBalance* message is drawn from the diagram frame to the *theirBank* lifeline. The *getBalance* message has a gate.

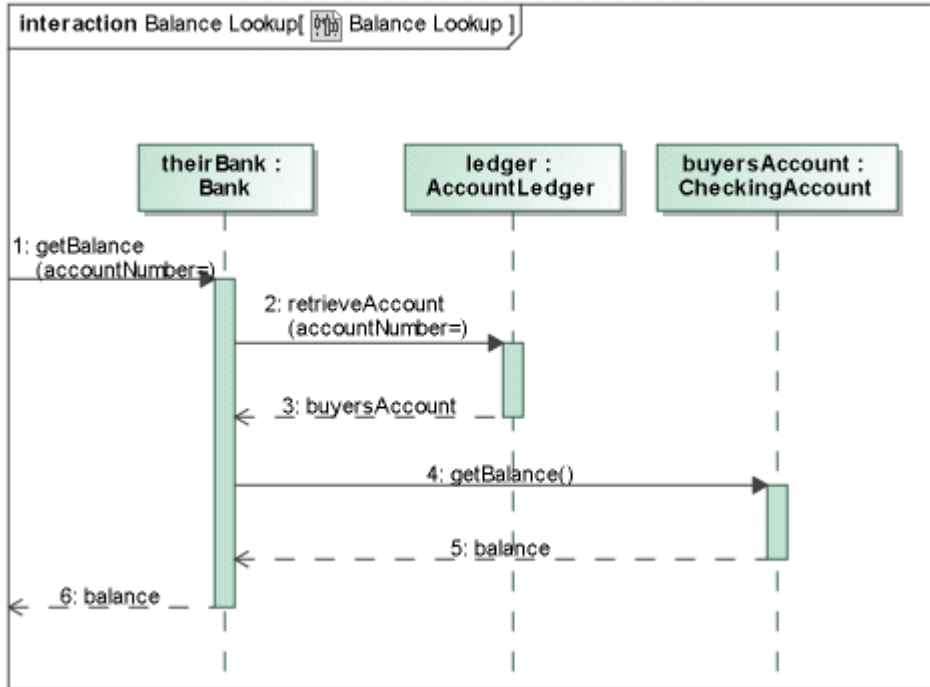


Figure 379 -- Formal gates usage in Sequence diagram

See Figure 380 on page 898 where the actual gate is presented. The *Balance Lookup* interaction use refers to the *Balance Lookup* sequence diagram. The *getBalance* message (see the 2nd message) has selected the formal gate and automatically repeats the data of the *getBalance* message from the *Balance Lookup* diagram.

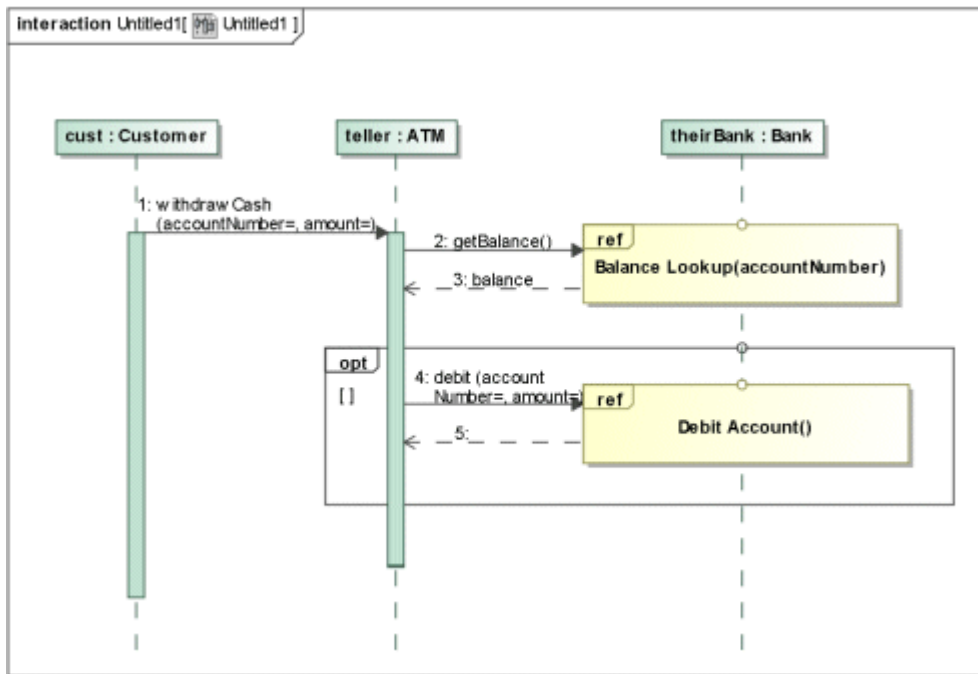


Figure 380 -- Actual gates usage in sequence diagram

Generalization

Term	<p>The OMG UML specification (UML 2.2: Superstructure) states:</p> <p>“A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier.”</p> <p>“Where a generalization relates a specific classifier to a general classifier, each instance of the specific classifier is also an instance of the general classifier. Therefore, features specified for instances of the general classifier are implicitly specified for instances of the specific classifier. Any constraint applying to instances of the general classifier also applies to instances of the specific classifier.”</p>
-------------	--

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the generalization in the **Generalization Specification** dialog box.

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set a general information about the generalization relationship.	General	The name of the parent element.
	Specific	The name of the child element.

To group generalization paths into a tree

1. Draw a generalization path between the child element and the parent element.
2. Click the generalization path button on the diagram toolbar.
3. Click the other child shape.
4. Drag the path to the other generalization path and drop it there.
 - Select **Make Sub Tree** from the parent class shortcut menu.
 - Select **Make Sub Tree** from the parent package shortcut menu.

To ungroup a generalization tree

1. Click the generalization tree's hollow triangle pointing to a parent element.
2. From the tree shortcut menu, select **Ungroup Tree**.

To separate the generalization path from the generalization tree

- From the generalization path shortcut menu, select **Remove From Tree**.
- Drag one generalization path to another class.

NOTE This command is available if the same tree contains the selected generalization path.

Generalizable elements

A generalizable element is a model element that may participate in a generalization relationship.

Name	Function
Is Abstract	Specifies whether the generalizable element may or may not have a direct instance. True indicates that an instance of the generalizable element must be an instance of a child of the generalizable element. False indicates that there may be an instance of the generalizable element that is not an instance of a child. An abstract generalizable element cannot be instantiated since it does not contain all the necessary information.
Is Leaf	Specifies whether the generalizable element is with descendants. True indicates that it may not have any descendant. False indicates that it may have some descendants (whether or not it actually has any descendants at the moment.)

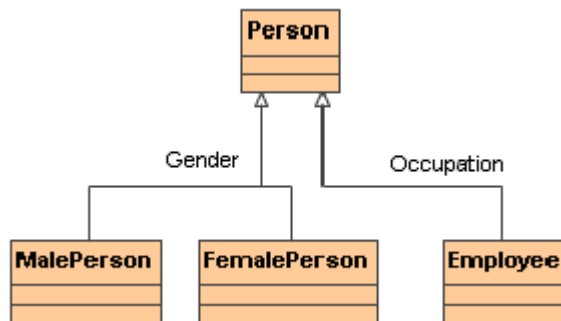
To define a generalizable model element (class, package, use case, etc.) as abstract or leaf

1. Open the corresponding **Specification** dialog box.
2. Select the **Is Abstract** or **Is Leaf** check box(es) in the **General** tab.

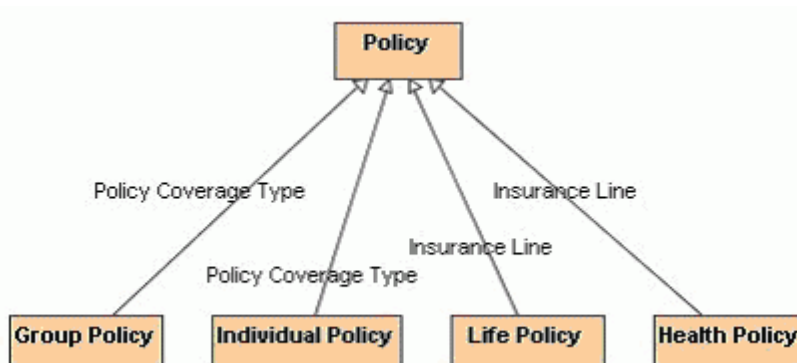
Generalization sets

Each generalization is a binary relationship that relates classifier to a more specific classifier. The Generalization Set defines a particular set of generalization relationships that describes the way in which a general classifier may be divided using specific types.

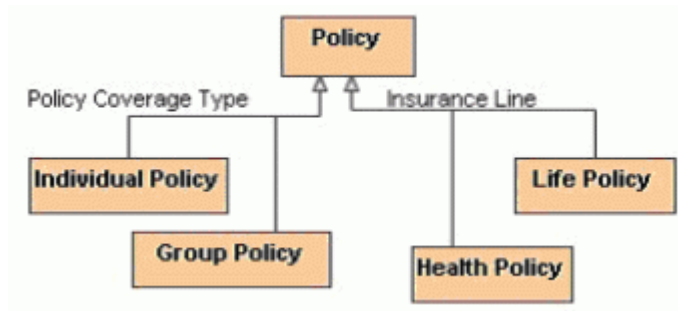
For example: the Person class can be specialized as either a Male Person or a Female Person. Furthermore, the Person may be specialized as an employee. The Female Person and Male Person constitute one Generalization Set and the Employee another:



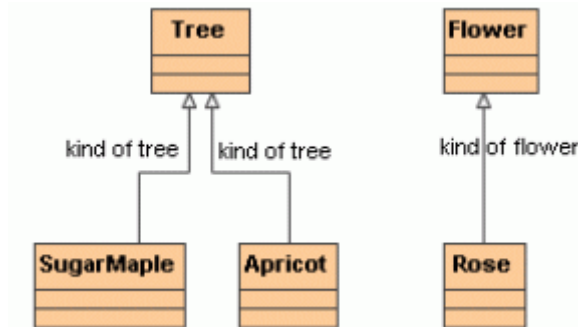
The label next to the generalization designates the name of the Generalization Set to which the generalization belongs.



The generalization set may be drawn as two or more generalization lines drawn to the same arrow-head. In this case it is labeled once. Drawing such a tree helps to understand the grouping of generalizations sets quickly .



The generalization set may contain only generalizations that have the same General element. The following is the example of generalizations, which may not have the same generalization set:



From the above example, the generalization, which is between the *Flower* and *Rose* classes, may not belong to the kind of tree generalization set, because its general element is *Flower* and the general element for the kind of tree generalizations is *Tree*. MagicDraw only lists all generalization sets that are allowed to assign. If a generalization set created in project does not have any generalizations, it may contain any generalization sets.

To create a new or assign an existing generalization set

- In the **Generalization Specification** dialog box, near the **Generalization Set** box click "...". In the open **Select Elements** dialog box assign an existing generalization set or click the **Create** button to create new generalization set.

NOTE!

In the Generalization Set list many generalization sets can be listed, but in this case on the diagram pane, near the generalization, only the first in the list is displayed.

- On the diagram pane, select a generalization. From the generalization shortcut menu, select the **Generalization Set** command. In the opened list, select the generalization set or click the **New** button and in the indicated place create a new Generalization Set to which the current generalization is assigned.
- On the diagram pane select some generalizations, which you want to assign to the generalization set. From the shortcut menu, select the **Generalization Set** command.

NOTE!

The Generalization Set command is enabled only if all selected generalizations have the same general element.

- Draw or move a generalization line to a generalization set tree or to a generalization that belongs to a generalization set. The newly created generalization is assigned to the same generalization set.

The Generalization Set dialog box

Specify the Generalization set in the **Generalization Set** dialog box.

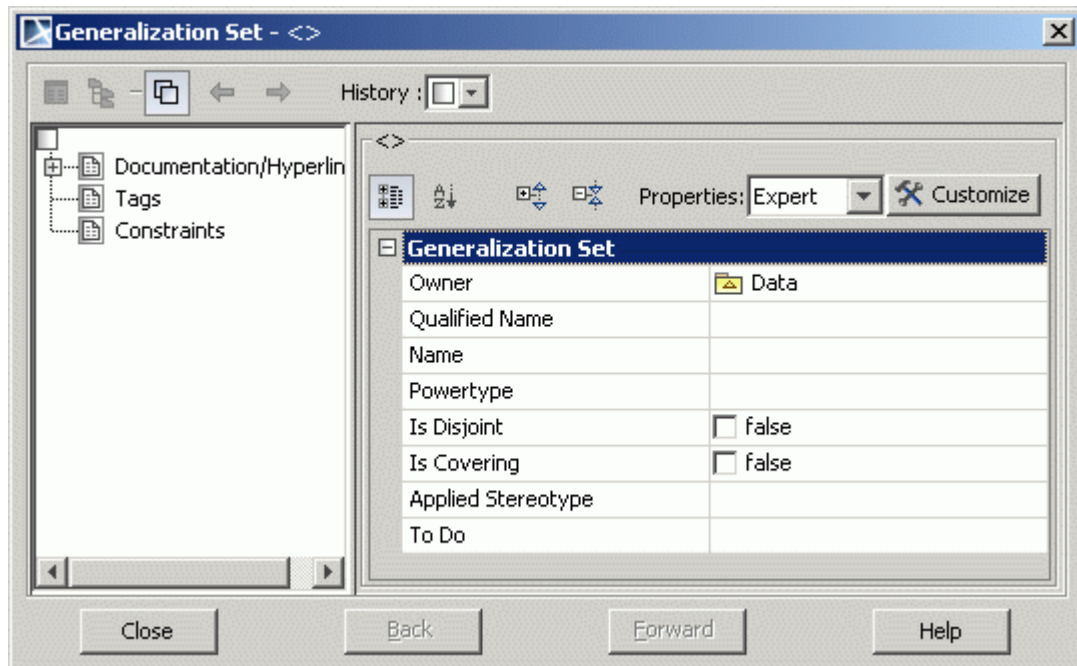


Figure 381 -- the Generalization Set dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set a general information about the generalization relationship.	Is Disjoint	Indicates whether or not a set of specific Classifiers in a Generalization relationship has a common instance. If the value is <i>true</i> , the Classifiers for a particular GeneralizationSet have no members in common; that is, their intersection is empty. The default value of the Is Disjoint property is false.
	Is Covering	Indicates whether or not a set of specific Classifiers is covering for a particular general classifier. When the value is <i>true</i> , every instance of a particular general Classifier is also an instance of at least one of its specific Classifiers for the Generalization Set.

In the **Generalization Set** dialog box, the **Generalization** list box lists the generalizations which are assigned to the current generalization set.

To assign a generalization:

1. Near the Generalization list click the "..." button. The list of available generalizations opens.
2. Near the generalization you want to assign to the generalization set, select the check box.
3. Click the **Apply** button.

NOTE!

The generalizations that belong to other sets are allowed to be selected. Selecting such a generalization removes it from the previous set and adds it to the current one.

To open the Generalization Set symbol properties

On the diagram pane select the generalization set name and from its shortcut menu, select the **Symbol(s) Properties** command.

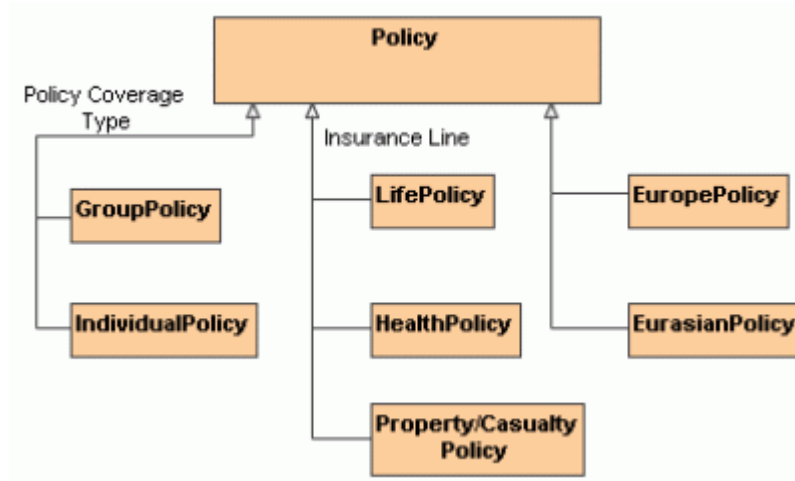
If the **Show Powertype** check box is selected, on the diagram pane, next to the generalization, the label of Powertype of Generalization Set is displayed instead of the Generalization Set name. The default **Show Powertype** property value is *false*.

If the **Show Complete/Disjoint** check box is selected, on the diagram pane, next to the generalization, the values of **Is Covering** and **Is Disjoint** properties are displayed on the diagram. The default **Show Complete/Disjoint** property value is *false*.

To group the generalizations to the generalization set trees

1. Select a general classifier on the diagram pane.
2. From the classifier shortcut menu, select the **Make Generalization Set Tree** command. The generalizations are grouped to trees, according to the generalization sets.

When the **Make Generalization Set Tree** command is selected, a tree is created for each generalization set. The generalizations, which do not belong to any generalization set, are grouped to a separate tree. For example:



Include

An include (uses) relationship from the use case A to the use case B indicates that an instance of the use case A will also contain the behavior as specified by B.

The include relationship is used when there are common parts of the behavior among two or more use cases. Each common part is then extracted to a separate use case, to be included by all base use cases having this part in common. Since the primary use of the include relationship is to reuse

the common parts, what is left in the base use case is usually not complete in itself but dependent on the included parts for meaning and context. This is reflected in the direction of the relationship, indicating that the base use case depends on the addition but not vice versa.

Define the extend relationships in the **Include Specification** dialog box.

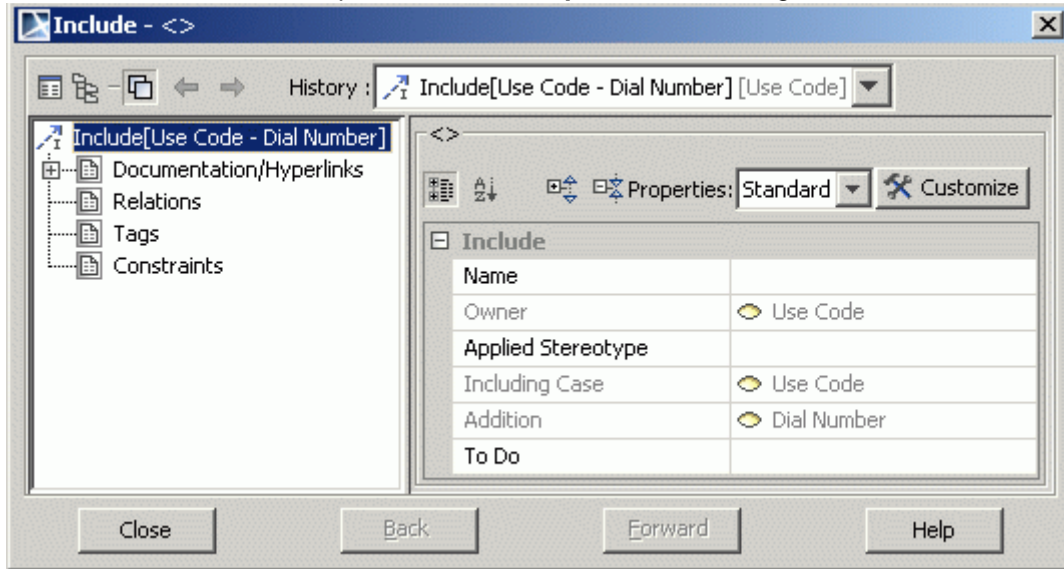


Figure 382 -- Include Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about specification elements not covered in this section.

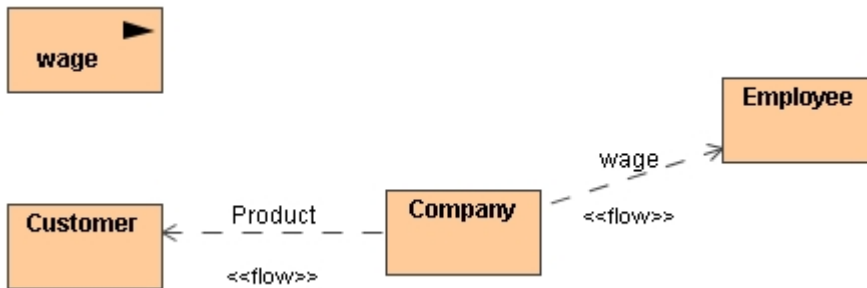
Tab name	Box	Function
General Set a general information about the include relationship.	Owner	The name of the use case, which owns the include relationship.
	Including Case	Name of the use case to where the relationship goes.
	Addition	Name of the use case from where the relationship comes.

Information Flow

An Information Flow specifies that one or more information items circulates from its sources to its targets. Information flows require some kind of “information channel” for transmitting information items from the source to the destination.

An information channel is represented in various ways depending on the nature of its sources and targets. It may be represented by connectors, links, associations, or even dependencies. For example, if the source and destination are parts in some composite structure diagrams such as a collaboration, then the information channel is likely to be represented by a connector between them. Or, if the source and target are objects (which are a kind of InstanceSpecification), they may be represented by a link that joins the two, and so on.

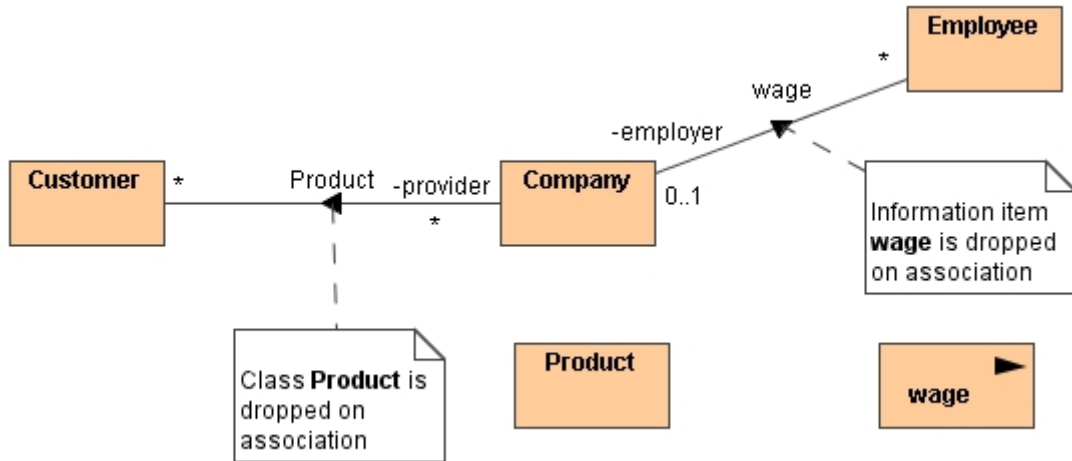
The information flow and the information item notation are added. You may draw them using the Information Flows toolbar in the class or composite structure diagram:



You can also create information flows in the associations in the class diagram and on the connectors in the composite structure diagrams:

3. Draw classes and associations.
4. From the association shortcut menu select command **Symbol(s) Properties** and select the **Show conveyed information A** and **Show conveyed information B** checkboxes.

5. Drag the class or information item on an association. An information flow is created.



Information Item

The InformationFlows package provides mechanisms for specifying the exchange of information between entities of a system at a high level of abstraction.

The Information flows describe a circulation of information in a system in a general manner. They do not specify the nature of the information nor the mechanisms by which this information is conveyed (message passing, signal, common data store, parameter of operation, etc.). They also do not specify sequences or any control conditions. It is intended that, while modeling in detail, representation and realization links will be able to specify which model element implements the specified information flow, and how the information will be conveyed.

An information item is an abstraction of all kinds of information that can be exchanged between objects. It is a kind of classifier intended for representing information in a very abstract way, the one which cannot be instantiated.

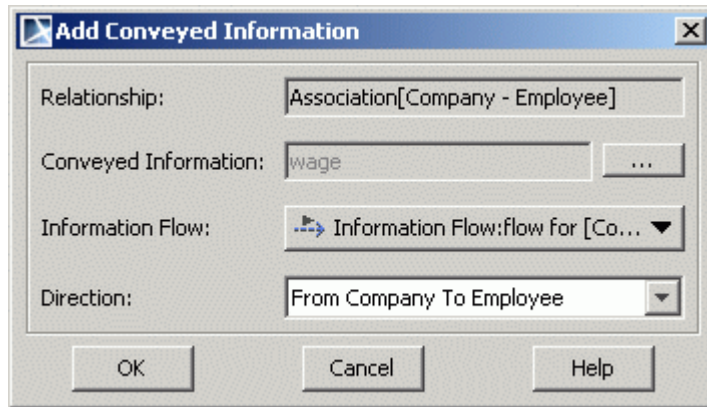
One purpose of information items is to be able to define preliminary models, before making a detailed modeling decisions on types or structures. Another purpose of information items and information flows is to abstract complex models by using a less specific but more general representation of the information exchanged between entities of a system.

In a classifier, the information item can be represented as a name inside a rectangle. The black triangle icon on top of this rectangle indicates that it is an information item.



the information Items (or any conveyed classifiers) can be displayed on any relationship:

1. Select the information item on the diagram pane and drag it on the relationship shape. The **Add Conveyed Information** dialog box opens.



2. After specifying information flow and direction arrow, click **OK**.

Instance

An instance specification specifies the existence of an entity in a modeled system and completely or partially describes the entity.

The description may include:

- The classification of an entity by one or more classifiers of which the entity is an instance. If the only classifier specified is abstract, then the instance specification only partially describes the entity.

- A kind of instance based on its classifier or classifiers - for example, an instance specification whose classifier is a class describes an object of that class, while an instance specification whose classifier is an association describes a link of that association.
- A specification of values of structural features of the entity. Not all structural features of all classifiers of the instance specification need to be represented by slots, in which case the instance specification is a partial description.
- A specification of how to compute, derive, or construct the instance (optional).

MagicDraw allows you to create the instances of classifiers – class, interface, enumeration, use case, actor, node, component, artifact, and other classifiers.

The instances are shown using a rectangle by underlining the name string of an instance element. The instance of an actor is shown as an actor “stick man” figure with the actor’s name string below the symbol.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the selected instance in the **Instance Specification** dialog box.

An instance is an individual unit with its own identity and value. Each instance has a descriptor – model element.

To open the **Instance Specification** dialog box

Select **Specification** from the instance shortcut menu or double-click the instance shape.

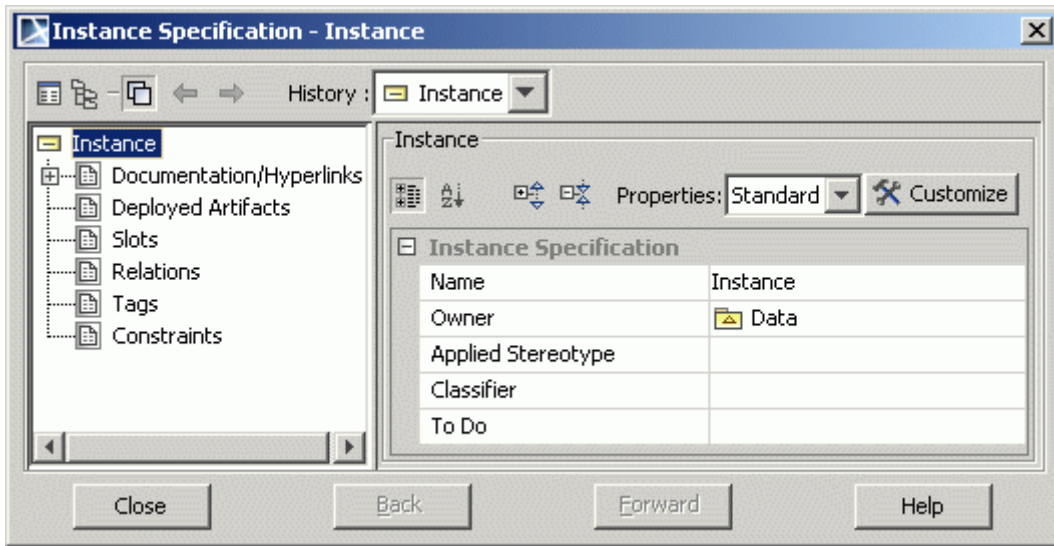


Figure 383 -- Instance Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General Set general options for the instance.	Owner	The name of the element, containing an instance.
	Classifier	The name of the assigned classifier.
Deployed Artifacts List of an artifacts or artifact instances that has been deployed to a deployment target.	Name	The name of the artifact or instance.
	Type	An element, which appears as a type of deployed artifact.
	Owner	The name of the element, containing deployed artifact.

Tab name	Box name	Function
Slots A named slot in an instance, which holds the value (the instance that is the value of the AttributeLink) of an attribute (the attribute from which the AttributeLink originates).	Type	The name, which is the owner of the attribute.
	Attribute	The name of the attribute.
	Instance	The name of an instance, to which the current instance is set as a default value.

To assign an already existing classifier to an instance

1. Right-click the instance shape and select **Type** from the instance shortcut menu.
2. Select the classifier you wish to assign to an instance.
 - Open the **Instance Specification** dialog box and in the **Classifiers** field, click the '...' button. The **Select Elements** dialog box opens. Move the classifier from **All** to the **Selected** list and click **OK**.

To assign/unassign an existing classifier for an instance in the Instance Specification dialog box

1. Double-click an instance shape or select **Specification** from the shape shortcut menu.
2. The **Instance Specification** dialog box appears. Click the '...' button in the **Classifiers** property.
 - To assign a new classifier, select an element in **All** and click the **Add** button to move it to the **Selected** list.
 - To unassign the assigned classifier, select an element in the **Selected** list and click the **Remove** button.

To hide/show an assigned classifier

From the instance shortcut menu, select/clear the **Show Classifier** check box.

To set the initial value to an attribute of the assigned classifier

1. Double-click an instance shape or select **Specification** from the shape shortcut menu.

2. The **Instance Specification** dialog box opens. Click the **Slots** tab.
3. Click the **Edit Value** button and type the name of the value.

To show/hide slots of the assigned classifier

Clear/select **Suppress Slots** check box in the instance shortcut menu.

NOTE

By default slots of the classifier are suppressed.

To display slot type on the instance symbol

Slot type can be optionally displayed on Instance or Part shapes.

Property **Show Slot Type** is added to slot symbol properties (select command **Symbol(s) Properties** from instance shortcut menu to invoke **Properties** dialog). Slot type name (see Figure 384 on page 914), slot type qualified name (see Figure 385 on page 914) or no slot type (see Figure 386 on page 915) may be displayed next to slot.

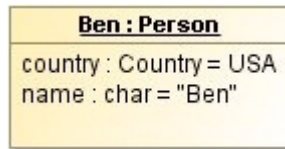


Figure 384 -- Slot type name is displayed next to slot



Figure 385 -- Slot type qualified name is displayed next to slot

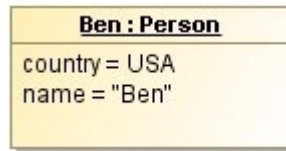


Figure 386 -- No slot type is displayed

To select slot in a diagram

Slot can be selected in a diagram. This allows the deletion of a slot straight from the diagram and to attach a note to a slot.

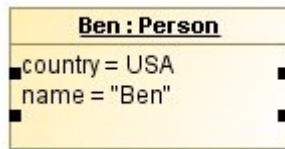


Figure 387 -- Slot selected in a diagram

Instance Specification

An instance specification represents an instance in a modeled system.

In the implementation diagram Node Instance, Component Instance, Artifact Instance elements are the same Instance Specification elements with an assigned component, node or artifact.

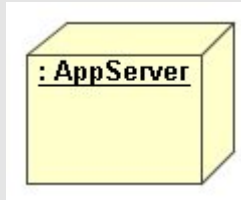
To create a Component Instance

1. In the Implementation diagram toolbar, click the Component Instance button. The **Select Components** dialog box opens.
2. Select a component from the list or click the **Create** button to create a new one. Click **OK**.

The same is valid for the Node Instance and Artifact Instance creation.

TIP!

Use the Node Instance button to create the Instance Specification with assigned Node and the Node Instance will have a Node shape.



To display specification value on the Instance Specification symbol

Specification value can be optionally displayed on the Instance Specification symbol. Check box **Show Specification Value** is added to Instance Specification properties (select command Symbol(s) Properties from instance shortcut menu to invoke **Properties** dialog).

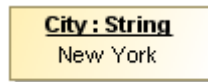


Figure 388 -- Specification value is displayed on the Instance Specification symbol

Assigning Instance Specification as Default Value quickly

You can drag Instance Specification element on Classifier property on a diagram to assign it as default value. Drag and drop is available only if Instance Specification classifiers are compatible with Property type and if Property is editable.

Interaction Use

Interactions are units of behavior of an enclosing Classifier. They focus on the passing of information with Messages between the Connectable Elements of the Classifier.

A reference to the interaction can be created.

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

To add a reference to an element

- In the **Interaction Use** specification dialog box, click the **Refers To** drop down list. A list of interactions, created in the project, opens. Click ‘...’ button, to add the interaction from the **Select Element** tree. Click the **Create** button to create a new interaction.
- From the interaction use shortcut menu select the **Refers To** command. In the appeared list select the interaction or create a new one.

To add an actual gate

1. Add a reference to the diagram, from which the diagram frame formal message is created.
2. To the current interaction use draw an actual message with the selected formal gate. For more information about working with gates, see “Gate” on page 894.

Interface

An interface is a specifier for the externally-visible operations of a class, component, or other classifiers (including subsystems) without a specification of the internal structure. Each interface often specifies only a limited part of the behavior of an actual class.

The set of interfaces realized by a classifier is its provided interfaces, which represent the obligations that instances of that classifier have to their clients. They describe the services that the instances of that classifier offer to their clients.

All options associated with an interface can be set in the **Interface Specification** dialog box.

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
Signal Receptions		Manage the receptions of an interface in the Signal Receptions pane. For more information about signal receptions, see “Reception” on page 997.
Inner Elements Add class, use case, interface, enumeration, data type, primitive, collaboration, or constraint to an interface	Name	The model element name.
	Type	the model element type.
	Create	Select an element from the list. The corresponding (class, use case, interface, enumeration, data type, primitive, collaboration, or constraint) specification dialog box opens. Define the selected model element in the dialog box.
	Delete	Remove the selected model element from an interface.

Provided and Required Interfaces

The set of interfaces realized by a classifier is its provided interfaces, which represent the obligations that instances of that classifier have to their clients. They describe the services that the instances of that classifier offer to their clients.

The interfaces may also be used to specify required interfaces, which are specified by a usage dependency between the classifier and the corresponding interfaces. Required interfaces specify services that a classifier needs in order to perform its function and fulfill its own obligations to its clients.

To draw a Provided Interface

1. In the Class diagram toolbar, select the Interface Realization path to draw from a class to an interface.

2. Suppress the attributes and operations of the interface (from the interface shortcut menu, **Presentation Options** submenu, select the **Suppress Attributes** and **Suppress Operations** check boxes.).

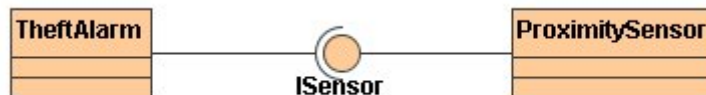


To draw a Required Interface

1. In the Class diagram toolbar, select the Usage path to draw from a class to an interface.
2. Suppress the attributes and operations of the interface (from the interface shortcut menu, **Presentation Options** submenu, select the **Suppress Attributes** and **Suppress Operations** check boxes.).



To draw both Provided and Required Interfaces together



Working with s is similar to working with classes. For more information, see “Working with classes” on page 855.

A general information about working with shapes is offered in Chapter 5, “Working With Diagrams.”

Provided and Required Interfaces in the Composite Structure diagram

Term The OMG UML specification (UML 2.2: Superstructure Specification) states:

“Ports represent interaction points between a classifier and its environment. The interfaces associated with a port specify the nature of the interactions that may occur over a port. The required interfaces of a port characterize the requests that may be made from the classifier to its environment through this port. The provided interfaces of a port characterize requests to the classifier that its environment may make through this port.”

Information about provided and required interfaces is crucial in the assembly stage of complex internal structures. It helps to decide where connectors should be attached.

Provided and required interfaces are valuable parts of the UML Composite Structure Diagram and SysML Internal Block Diagram.

A provided interface is shown using the "lollipop" notation attached to the port and required interface is shown using the "fork" notation attached to the port (see Figure 389 on page 920).

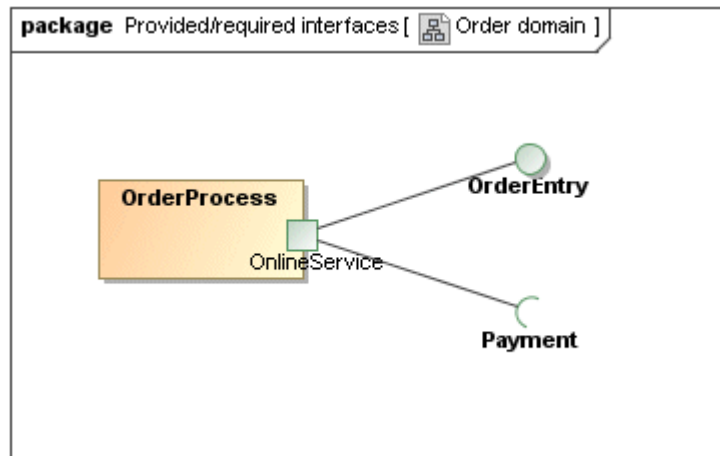


Figure 389 -- Provided and Required interface

In the Composite Structure diagram you cannot draw provided and required interfaces itself, but with the new functionality of MagicDraw you can display preexisting port with the required and provided interfaces as images.

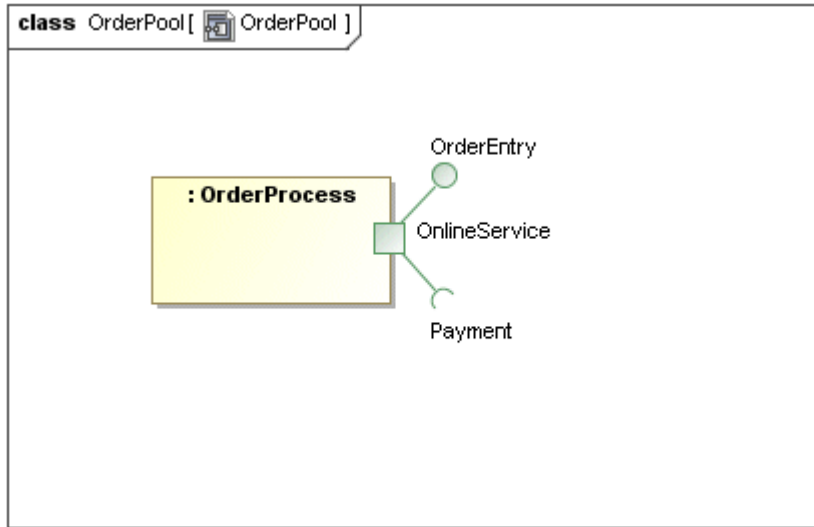


Figure 390 -- Provided and required interface in the Composite Structure diagram

Lollipop and fork symbols in the Composite Structure diagram are implemented as small attachments to a Port symbol (like name label). It is not the same as the independent standalone notation of the interface, it is only part of port symbol. It is important for Composite Structure diagrams where real Interfaces (as Classifiers) can not be used.

It is an optional notation, a port does not display provided or required interfaces by default.

Displaying provided/required interfaces in the Composite Structure diagram

1. Create provided and required interfaces in the Class diagram. See section "To draw a Provided Interface" on page 918 and section "To draw a Required Interface" on page 919.
2. In the Composite Structure diagram, select **Related Elements** from the port shortcut menu and then **Display Provided/Required Interfaces**. Or, in the individual Port symbol **Properties** dialog box, select the **Show Provided Interfaces** and **Show Required Interfaces** check boxes.

As Port can provide or require many interfaces, displayed or hidden interfaces can be managed in the **Edit Compartment** dialog box.

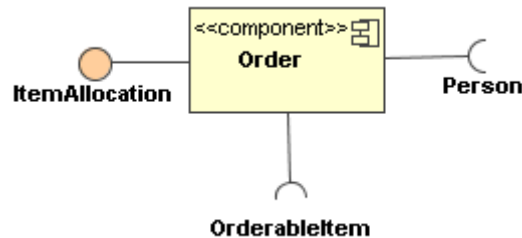
Provided/required interfaces in the Component diagram

A component specifies a formal contract of the services that it provides to its clients and those that it requires from other components or services in the system in terms of its provided and required interfaces.

The required and provided interfaces may optionally be organized through ports.

To add and manage the added provided and required interfaces quickly, in the **Component Specification** dialog box, select the **Provided/Required Interfaces** pane.

For more information about provided and required interfaces, see “Provided and Required Interfaces” on page 918.



Internal transition

In all other cases, the action label identifies the event that triggers the corresponding action expression. These events are called internal transitions and are semantically equivalent to self transitions except that the state is not exited or re-entered. This means that the corresponding exit and entry actions are not performed.

For more information on defining transitions, see “The join vertices are used to merge several transitions emanating from the source vertices in different orthogonal regions. The transitions entering a join vertex cannot have guards or triggers.” on page 994.

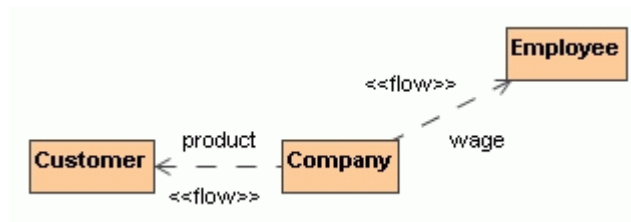
Specify the internal transition in the **Transition Specification** dialog box. For a detailed description of this dialog box, see “Transition Specification dialog box” on page 1017.

To define an internal transition

1. Double-click the state or select **Specification** from the state shortcut menu. The **State Specification** dialog box opens.
2. Click the **Internal Transitions** group.
3. Click the **Create** button. The **Transition Specification** dialog box opens. Specify an internal transition.

To remove the internal transition, click the **Delete** button.

The Information Flow can be related to any relationship.



Lifeline

A lifeline represents an individual participant in the Interaction. The lifeline represents only one interacting entity. It is shown using a rectangle symbol.

For more general information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the lifeline in the **Lifeline Specification** dialog box.

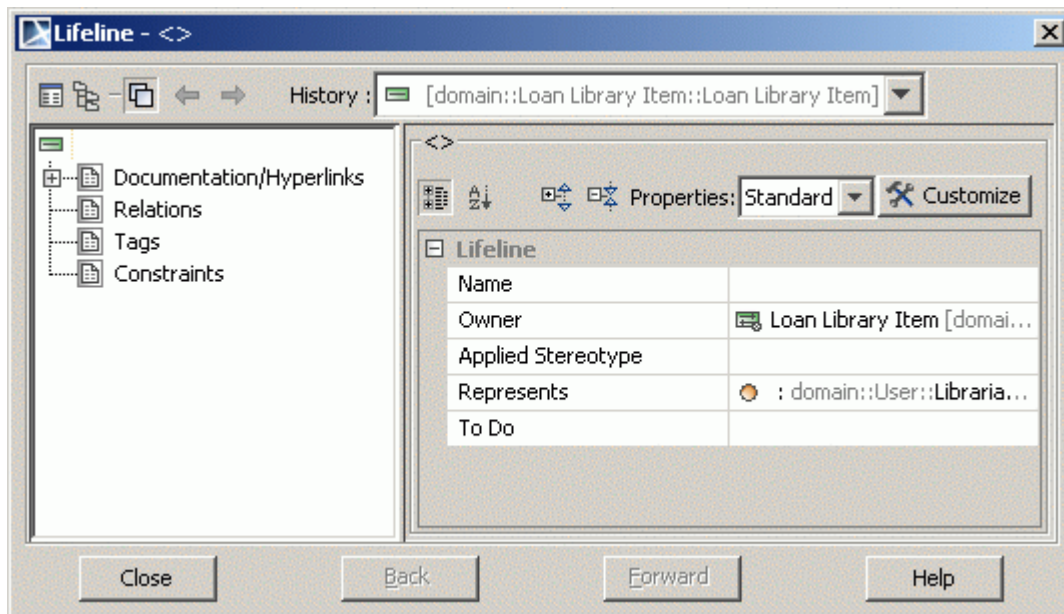


Figure 391 -- Lifeline Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set a general information about the lifeline.	Owner	The name of the interaction, which owns a lifeline.
	Represents	The property name, which represents a lifeline in an interaction.

To assign a type (classifier) to a lifeline

1. Right-click the lifeline shape and select **Type** from the shortcut menu.
2. Select the classifier you wish to assign to a classifier role or click **New** and select the element from the list.

To hide/show a base classifier

Select the **Show Classifier** check box from the lifeline shortcut menu.

Lifeline in the Sequence Diagram

A lifeline represents the existence of the object at a particular time. It stretches from the top to the bottom of the diagram. In the sequence diagram, an object lifeline denotes an Object playing a specific role.

For more information about the **Lifeline Specification** dialog box, see “Lifeline Specification dialog box” on page 924.

To destroy a sequence object - a large “X” at the end of its lifeline marks its destruction

- Select **Destroy Mark** from the lifeline shortcut menu.
- Set a message as destroyed.

When an object receives a message, an activity starts in that object. An activation (focus of control) shows the period during which an object is performing an action either directly or through a subordinate procedure. The activation bar is used to denote that activity.

To change the activation bar size

1. Click the desired activation bar on the Diagram pane.
2. Drag the activation bar to the desired direction.

NOTE

After resizing, the lines on the activation bar are thickened, but the size may not change automatically.

To add a recursive message to a lifeline

Click the Recursive Message button on the diagram toolbar and click the lifeline on the desired place where you wish to draw this message.

Link

Link is instance specification with assigned classifier - association.

An instance specification whose classifier is an association represents a link and is shown using the same notation as for an association, but the solid path or paths connect instance specifications rather than classifiers.

End names can adorn the ends. Navigation arrows can be shown, but if shown, they must agree with the navigation of the association ends.

You can show role names and navigability on the link symbol (see Figure 392 on page 926). Properties **Show End A**, **Show End B**, and **Show Navigability** are added to link shortcut menu and symbol properties. These properties are displayed according classifier association.

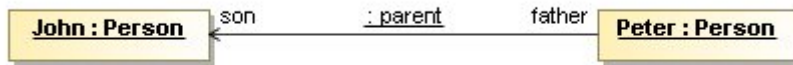


Figure 392 -- Role names displayed on the link symbol

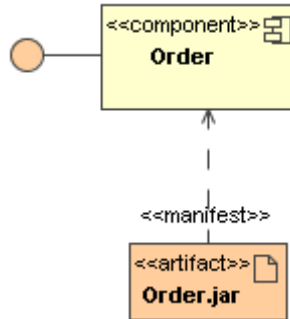
Manifestations

An artifact embodies or manifests a number of model elements. It owns the manifestations, each representing the utilization of a packageable element.

To create the manifestations, simply draw the Manifestation link from an artifact to a component.

To display the manifested artifacts on the component shape

From the component shortcut menu select **Presentation Options** and then clear the **Suppress Artifacts** check box.


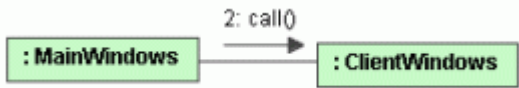
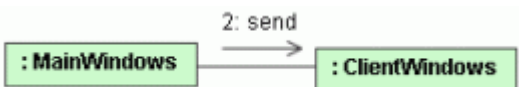
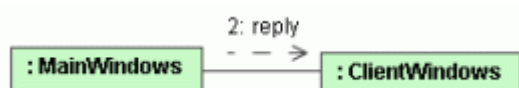
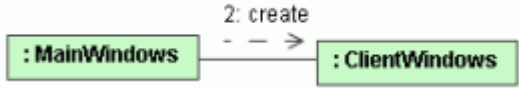



Message

A Message is an element that defines one specific kind of communication in an Interaction. A communication can be, for example, raising a signal, invoking an Operation, creating or destroying an Instance. The Message specifies not only the kind of communication given by the dispatching Execution Specification, but also the sender and the receiver.

A message is shown as a line from the sender message end to the receiver message end. The form of the line or arrowhead reflect the properties of the message:

Name	Function	Notation in Communication Diagram	Notation in Sequence Diagram
Asynchronous Message	The sender is not waiting for the recipient's acceptance.		

Name	Function	Notation in Communication Diagram	Notation in Sequence Diagram
Synchronous Message	The operation that should be completed before the caller resumes the execution.		
Call Message	A call message represents the request to invoke a specific operation.		
Send Message	A send message specifies the sending of a request to invoke a specific operation.		
Reply Message	The reply message returns the values to the caller of the previous call, completing the execution of the call.		
Create Message	A create message specifies the creation of a specific operation.		The message is connected directly to an object (not lifeline).
Destroy Message	Destroy message represents the destruction of the instance described by the lifeline.		A large X mark is displayed on the object's lifeline in the message's destination.

To open the **Message Specification** dialog box

- Select **Specification** from the message shortcut menu or double-click the message.
- Open the **Association Role Specification** dialog box. Open the **Messages** tab and click the **Edit** button.

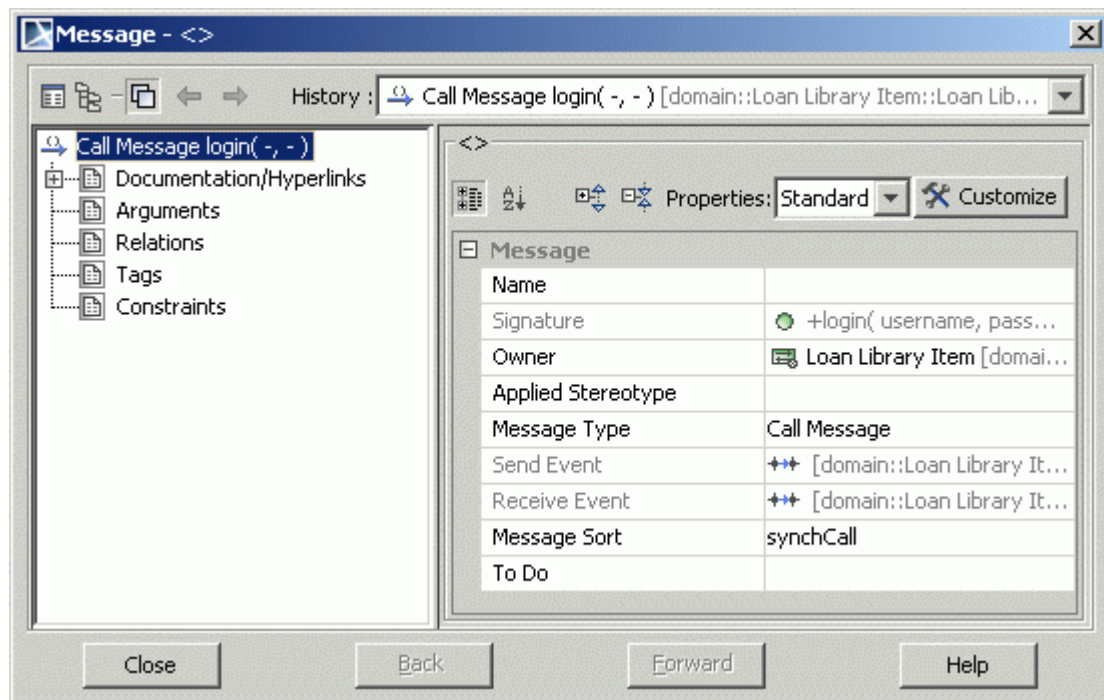


Figure 393 -- Message Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General Set a general information about the message.	Signature	The definition of the message sort, depending on its type. Note that this field is not editable.
	Owner	The name of the interaction, which owns the message.
	Send Event	The name of the sender interaction.

Tab name	Box name	Function
	Receive Event	The name of the receiver interaction.
	Message Sort	The type of the action: synchCall, asynchCall, asynchSignal, . Click the '...' button to select a type from the list: <ul style="list-style-type: none"> • synchCall • asynchCall • asynchSignal • createMessage • deleteMessage • reply
	Create	A list of arguments opens. Select an argument and specify the value in the Argument field.
	Delete	Removes an argument from the list.
	Up	Move the selected item to an upper position.
Arguments Specify a message argument value that appears during the execution.	Down	Move the selected item to a lower position.

To draw a message on the connector

1. Click the **Message to Right** or **Left** button on the diagram toolbar.
2. Click the desired connector on the Diagram pane. A message arrow is placed to the connector you click on the diagram.

NOTE

A message flow has two directions: right and left. Choose one of them by clicking the associated button on the diagram toolbar.

To set an action type for a message

1. Open the **Message Specification** dialog box.
2. Click the **Message Sort** "..." button and select the action type from the drop-down list. See "Message" on page 927.

To show/hide the numbers of the messages

From the communication diagram shortcut menu, select **Numbering** and then select/clear the **Show Meesage Numbers** check box.

To remove the automatic advanced numbering of messages

From the communication diagram shortcut menu, select **Numbering** and then clear the **Advanced Message Numbering** check box and you will be able to number messages according your needs.

To change the numbering of the messages

1. From the communication diagram shortcut menu, select the **Change Numbering** command.
2. Increase, decrease, and/or change the level of numbering in the **Change Communication Numbering** dialog box:

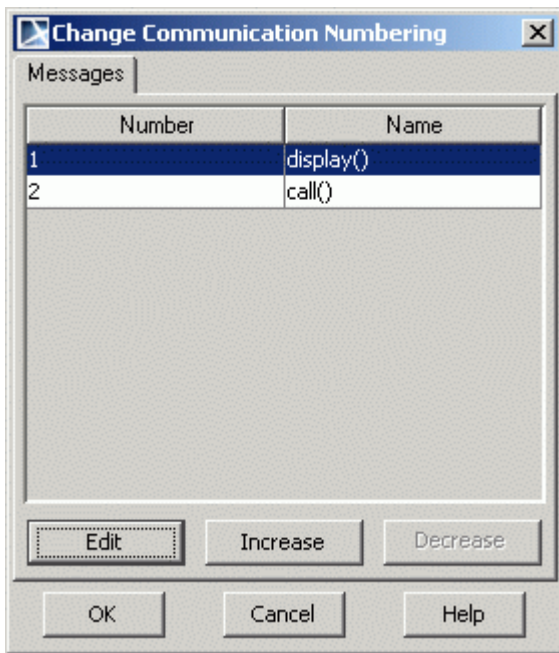


Figure 394 -- Change Communication Numbering dialog box

Box name	Function
Number	The number of the message.
Name	The type, name of the action of the corresponding message.
Edit	The Type Number dialog box opens. Type the number of the message.

Box name	Function
Increase	Increases the selected number to one point.
Decrease	Decreases the selected number to one point.
OK	Saves changes and exits the dialog box.
Cancel	Exits the dialog box without saving changes.
Help	Displays the MagicDraw Help.

Assigning/Creating operation for message

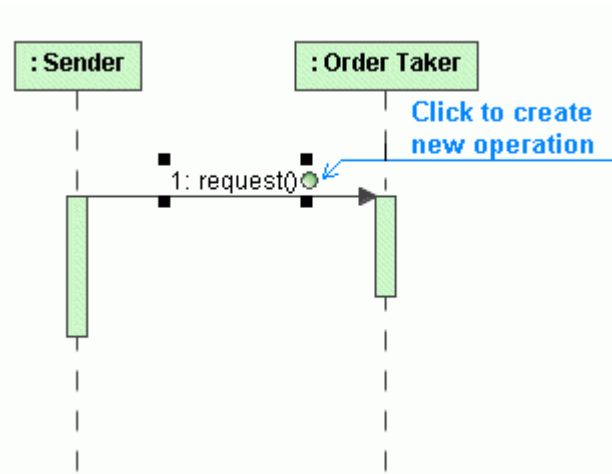
Usually a model creation order is the following: there is a created class diagram with classes and then the sequence diagram uses the classes and operations to representing the call order.

MagicDraw allows a faster way to assign or create operations than the traditional model creation does. A sequence diagram is created to represent classes and messages, and with single click you can convert a message into a call as well as create operations in the classes according to the message name.

To create a new operation to the message

1. Draw call message between the lifelines.

2. Select it on the diagram pane. Click on the smart manipulator at the end of the message name to create a new operation.



3. The new operation, the name of which is as same as that of the message, is created. The parameters, if provided, are added to the operation.

-or-

- From the message shortcut menu, select the **Create New Operation** command.

NOTE: The possibility to create a new operation exists only if the lifeline to which the message is drawn has an assigned type. The type element is not a read only type and it may have operations.

Assigning/Creating signal reception for message

To create a new signal reception to the message

1. Draw any send message between the lifelines. Lifelines need to have types specified.
2. Assign signal to the message.
3. Select message on the diagram pane. Click on the smart manipulator at the end of the message name to create a new signal reception.
4. The new signal reception, the name of which is as same as that of the message, is created for a type of a lifeline. The parameters, if provided, are added to the signal reception.

-or-

- From the message shortcut menu, select the **Create New Signal Reception** command.

Predecessors and activators

The predecessor is the set of messages of which the completion enables the execution of the current message. It is a comma-separated list of sequence numbers followed by a slash

('/'): sequence-number ',' . . . '/'

The meaning of the predecessor is that the execution of a message is not enabled until all of the communications of which the sequence numbers appeared in the list have occurred. Therefore, the list of predecessors represents a synchronization of threads. The message corresponding to the numerically preceding sequence number is an implicit predecessor and does not need to be explicitly listed.

All of the sequence numbers with the same prefix form a sequence. The numerical predecessor is the one in which the final term is one less. That is, number 3.1.4.5 is the predecessor of 3.1.4.6, where the number "3" is an activator.

To show the predecessors on the message

- Select **Show Predecessors** from the current message shortcut menu.
- Select the **Symbol(s) Properties** command from the message shortcut menu and select the **Show Predecessors** check box.

To change the activator number of the messages

1. Select the **Activator** command from the current message shortcut menu.
2. Select the number of activators that you want to assign for the current message and predecessors.

NOTE: If you change the activator number to one of the predecessor message, this number also changes for other predecessors.

Define the selected message in the **Message Specification** dialog box. For a detailed description of this dialog box, see Section "Message Specification dialog box" on page -929.

Message in Sequence Diagram

Messages are the main tool for displaying an interaction between objects, which is usually an operation call or a creation and destruction event. Messages are labeled with the name (operation or signal) and their argument values.

Define the selected message in the **Message Specification** dialog box. For more information on working with messages, “Message” on page 927.

Creating nested activation

Nested activations allow modeling parallel execution of operations that belong to a single class and modeling callback messages.

To create a nested activation, create at least two incoming messages for the activation, select the second (or any subsequent) message and from its shortcut menu select **Create Nested Activation**.

Nested activations can be created for the following message sorts:

1. synchCall
2. asynchCall
3. asynchSignal

Nested activations are used in three cases:

1. Modeling of parallel execution for a non-active object.
 - Create at least two incoming messages for the activation.
 - Select the second (or any subsequent) message and select the **Create Nested Activation** menu item from the shortcut menu.

- A nested activation will be created for the selected message.

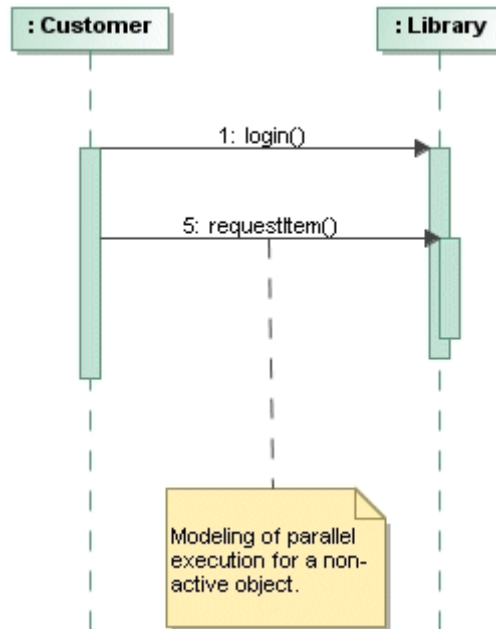


Figure 395 -- Nested activation

2. Modeling of parallel execution for an active object.
 - Create an outgoing message for the activation that has the "Show Entire Activation" mode enabled.
 - Select the created message and select the **Create Nested Activation** menu item from the shortcut menu.

- A nested activation will be created for the selected message.

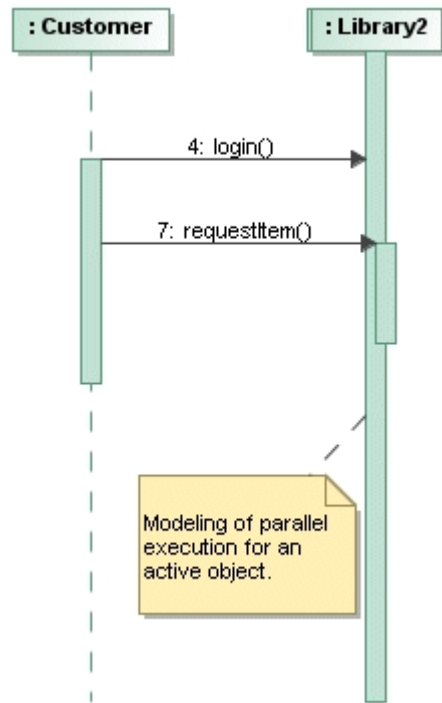


Figure 396 -- Parallel execution for an active object

3. Modeling of callback messages.

- Create an incoming message (the modeled callback message) for the activation that has an outgoing message to the activation for which the modeled message acts as an outgoing message.
- Select the created message and select the **Create Nested Activation** menu item from the shortcut menu.

- A nested activation will be created for the selected message.

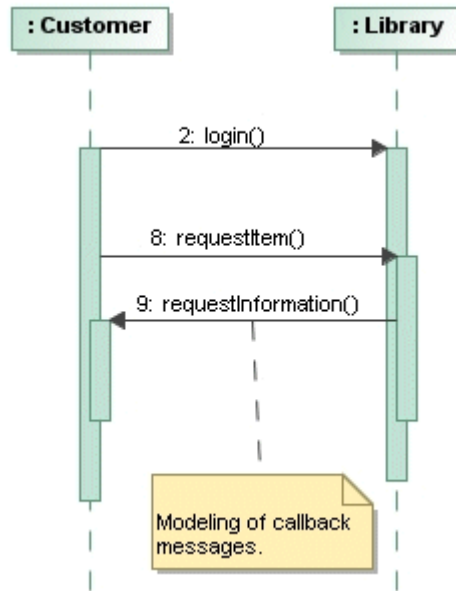


Figure 397 -- Callback messages

To merge the nested activations with parent activation

From the message shortcut menu select the **Reduce Nesting Level** command. The message will be connected to the parent activation.

Model

A model contains a (hierarchical) set of elements that together describe the physical system being modeled. It may also contain a set of elements that represents the environment of the system, typically Actors, together with their interrelationships, such as Associations and Dependencies.

A model is presented as a package with a small triangle in the upper right corner of the large rectangle. The triangle can be shown in the tab.

The model is defined as a package – it has package properties - in the **Model Specification** dialog box. For a detailed description of packages, see “Package” on page 955.

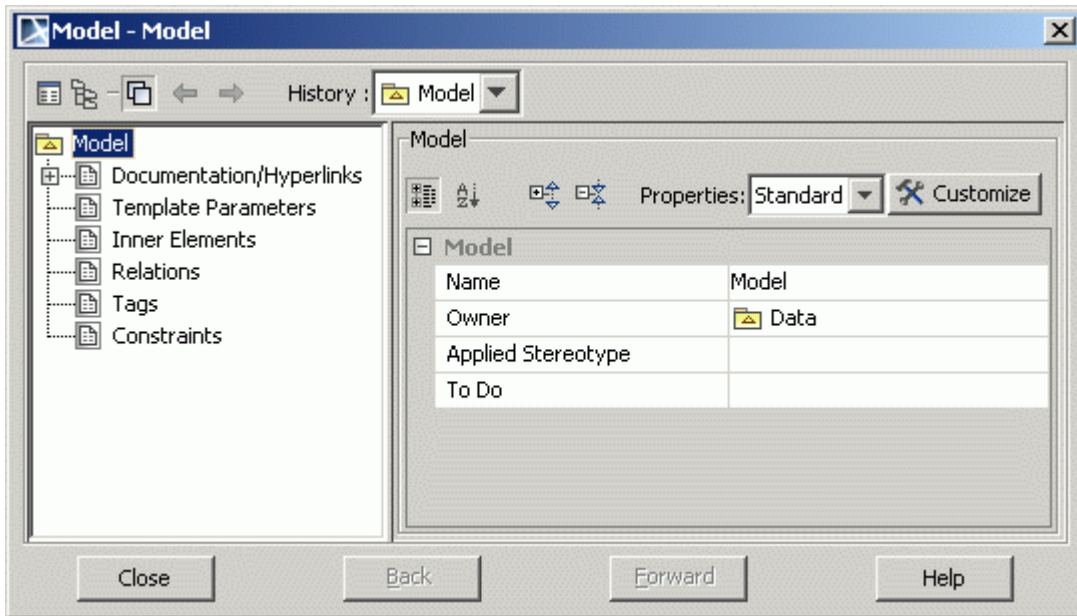


Figure 398 -- Model Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Node

Any computer or device that is relevant to the implemented system can be shown as a node. The node is drawn as a three-dimensional cube with a name inside it. Devices in a system are typically represented with a stereotype that specifies the device type. The nodes can be represented as types and as instances.

It is shown as a figure that looks like a 3-dimensional view of a cube.

For more information about working with symbols, see Chapter 5, “Working With Diagrams.”

Define the selected node in the **Node Specification** dialog box.

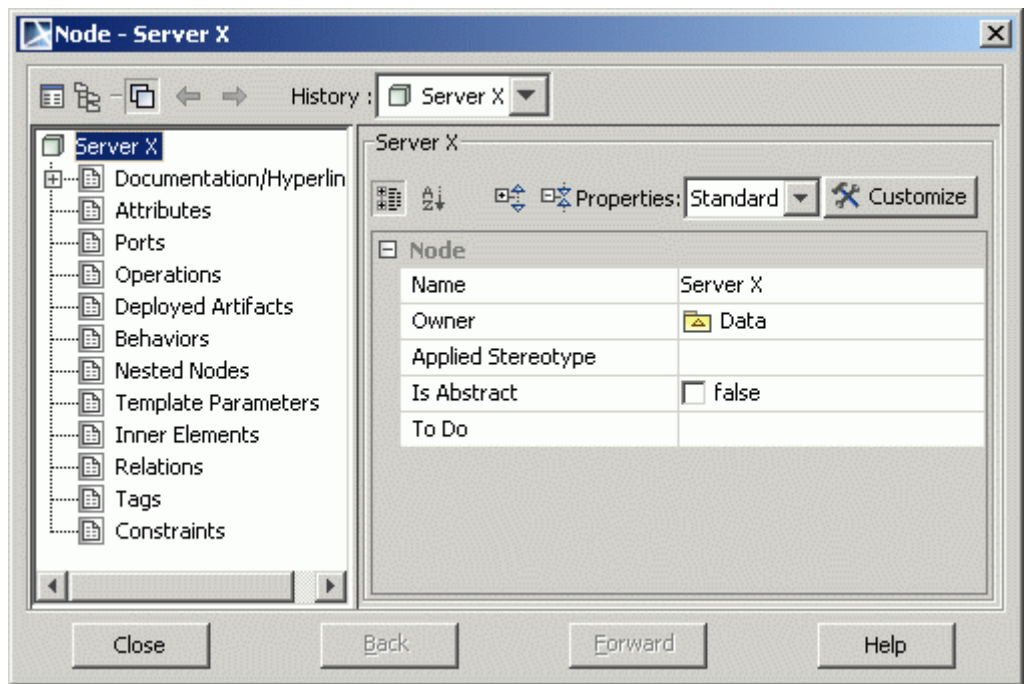



Figure 399 -- Node Specification dialog box

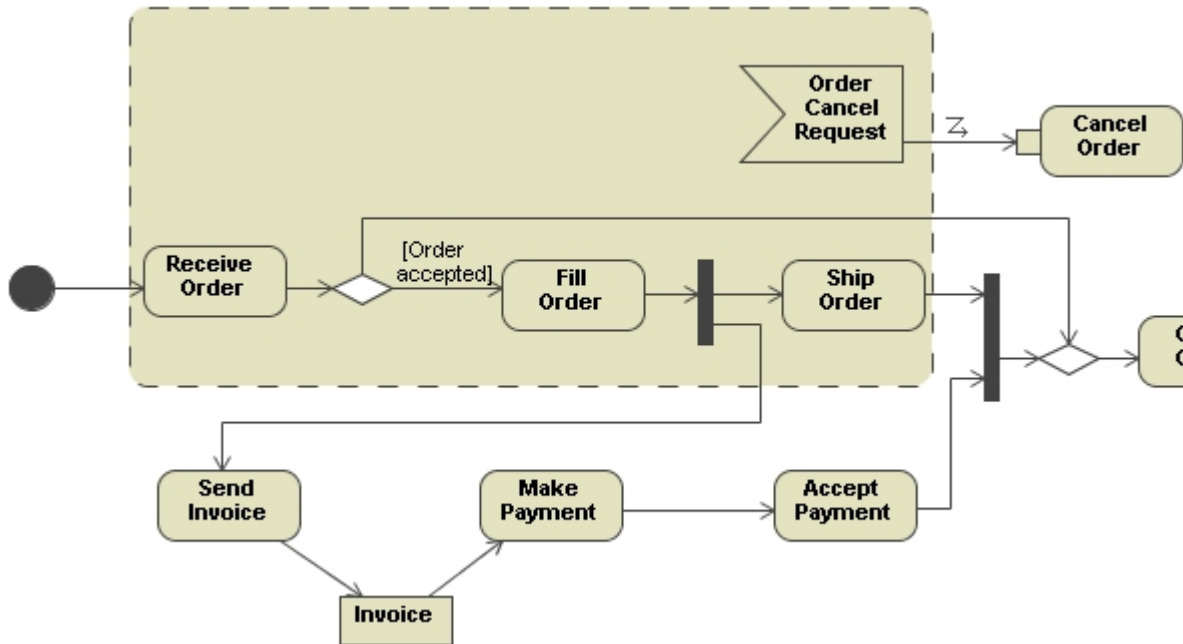
Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
Ports	Name	Name of the port.
	Type	Type, assigned to the port.
	Provided	The provided classifier is displayed.
	Required	The required classifier is displayed.
	Classifier	The name of classifier, owning a port.
	Create	Create a new port.
	Delete	Remove an existing port from the node.

Tab name	Box	Function
Deployed Artifacts	Name	Name of the deployed artifact.
	Type	Type of the item - artifact.
	Add	Add an artifact to the list.
	Remove	Remove an artifact from the list.
Behaviors	Name	Name of the behaviors.
	Type	Type, assigned to the behavior.
	Create	Select an item from the list - activity, interaction, and state machine.
	Delete	Remove a behavior from the class.
Nested Nodes	Name	Name of the nested node.
	Type	Type of the classifier (class, interface, etc.).
		Click  button to open the classifier Specification dialog box.
	Create	Select a node, device, or execution environment from the list. The element Specification dialog box opens.
Inner Elements Add another element to a node.	Remove	Remove a node from the list.
	Name	The model element name.
	Type	The model element type.
	Create	Select an element from the list. The corresponding specification dialog box opens. Define the selected model element in the dialog box.
	Delete	Remove the selected model element from the node.

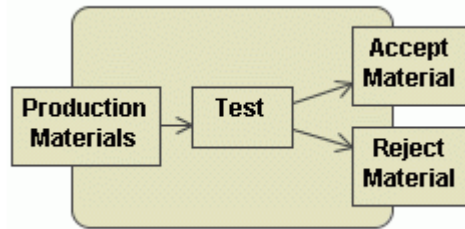
Structured activity node

A structured activity node is an executable activity node that may have an expansion into the subordinate nodes. It represents a structured portion of the activity that is not shared with any other structured node, except for nesting.



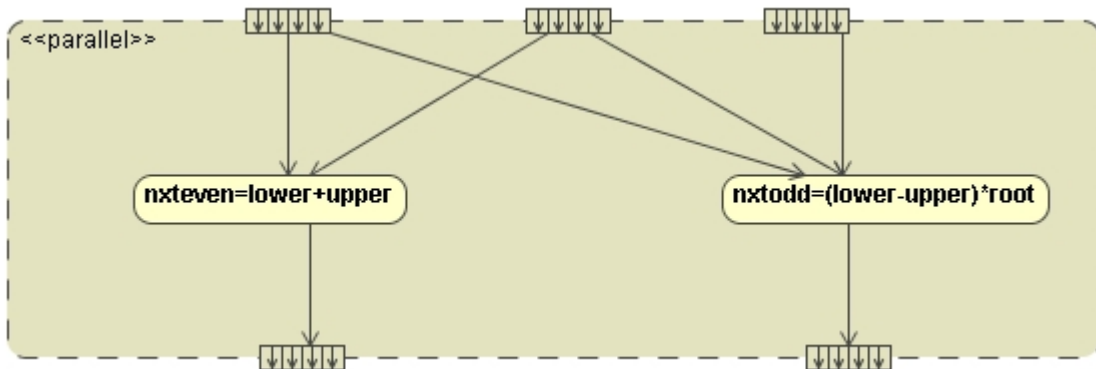
Activity parameter node

It is an object node for inputs and outputs to activities. The activity parameters are object nodes at the beginning and end of the flows, to accept inputs to an activity and provide outputs from it.



Expansion Region and Expansion Nodes

The Expansion Region and Expansion Nodes may be drawn in the activity diagram (the Input and Output Expansion Nodes may be found in the diagram toolbar Object Node button group):

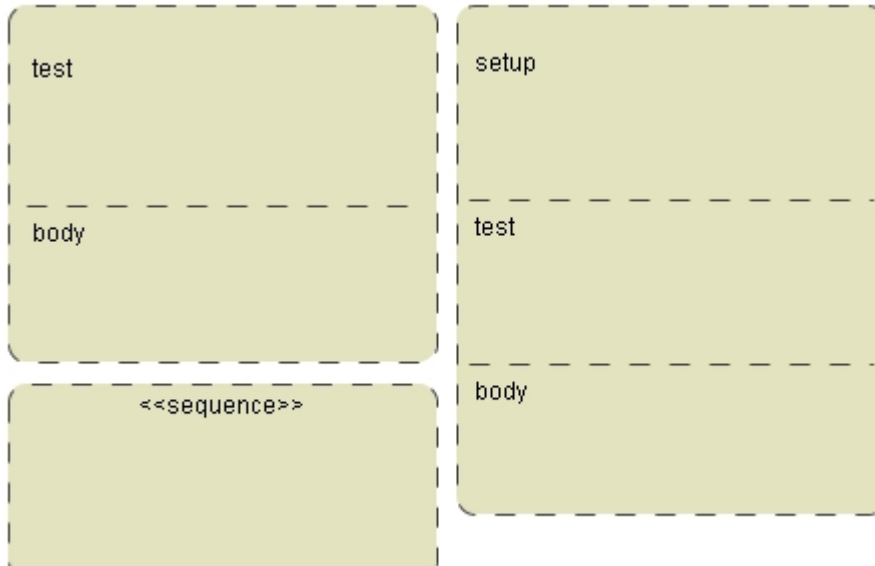


If, Loop and Sequence Conditional Nodes

A conditional node is a structured activity node that represents an exclusive choice among some number of alternatives.

A sequence node is a structured activity node that executes its actions in order.

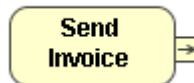
A loop node is a structured activity node that represents a loop with the setup, test, and body sections.



Object

To convert a pin to an object:

1. Draw an Input, Output, or Value Pin on the Action:



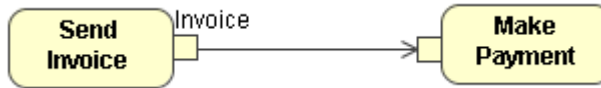
2. Select an Output, Input or Value Pin on the diagram pane and from its shortcut menu select **Convert to Object**. The Pin is converted to an Object:



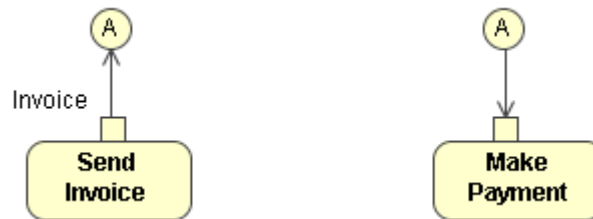
Object Flow

To split an Object Flow:

1. Draw an Object Flow relation between two Actions.



2. From the Object Flow shortcut menu, select the **Split Object Flow** command.



Object Node

An object node is an activity node that indicates an instance of a particular classifier, possibly in a particular state, may be available at a particular point in the activity. It can be used in a variety of ways, depending on where the objects are flowing from and to, as described in the semantics section.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the selected object node in the **Central Buffer Node Specification** dialog box.

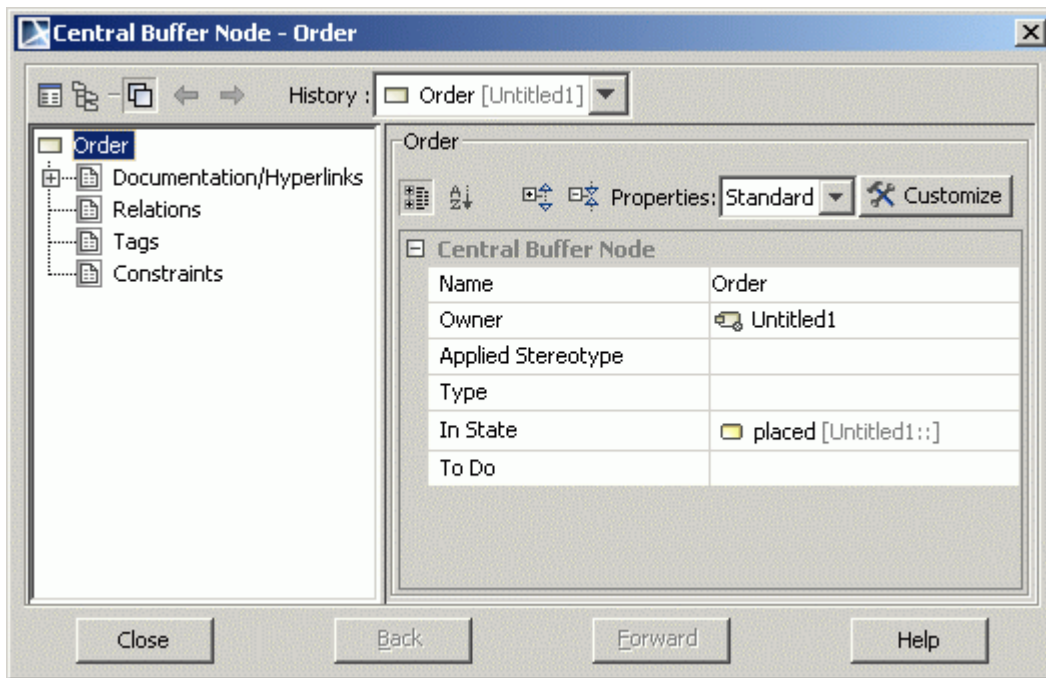


Figure 400 -- Central Buffer Node Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General Set a general information about the object node	Type	A list of classifiers. Select a classifier you wish to assign to an object node shape. Click the ‘...’ button to open the Select Element dialog box for selecting the type from the elements tree.
	In State	Click the “...” button to assign an existing final state or state from the model in the Select Elements dialog box, or click Create for defining a new one.

To set a classifier to an object node

1. Double-click the object node or select **Specification** from the shape shortcut menu. The **Central Buffer Node Specification** dialog box opens.
2. Select a classifier you wish to assign to an object node from the **Type** drop-down list.

To assign a state or final state to an object node

1. Click the “...” button in the **Central Buffer Node Specification** dialog box, **In State** field. The **Select Elements** dialog box opens.
2. Select a state from the existing model elements, or click **Create**. The **State Specification** dialog box opens. Specify a new state, which will be assigned to an object node.

Opaque Behavior

A behavior with implementation-specific semantics. The Opaque Behavior is introduced for implementation-specific behavior or for use as a place-holder before one of the other behaviors is chosen.

To create a new Opaque Behavior:

1. In the Browser, select a package.
2. From the package shortcut menu, select **New Element** and then **Opaque Behavior**. Enter the name for a newly created element.

To create an Opaque Behavior symbol

Drag and drop the selected Opaque Behavior element from the Browser tree on the Diagram pane.

Operation

Entries in the operation compartment are strings that show the operations defined on classes as well as those that are supplied by the classes. An operation is a service that can be requested to perform by an instance of the class. It has a name and a list of arguments.

Usually class attributes are accessed through the operations. The operations are used to perform specific actions, such as system calls, utility functions, and queries. The operation signature provides all information needed to use that operation.

To create a new operation

- Double-click the selected class or select **Specification** from the class shortcut menu. The **Class Specification** dialog box opens. Click the **Operations** tab and then click the **Create** button. The **Operation Specification** dialog box opens. Define a new operation and click **OK**.
- Select **Insert New Operation** from the class shortcut menu. Type an operation name directly on the class shape.
- In the Browser tree, select an already created class. From the class item shortcut menu, choose **New** and then **Operation**.
- Select a class shape, press CTRL+ALT+O shortcut key and type the operation name on the Diagram pane.
- Select the class shape and click the small green **Insert New Operation** smart manipulation button.

Define an operation in the **Operation Specification** dialog box.

To open the **Operation Specification** dialog box

1. Double-click the class /actor or select **Specification** from the class/actor shortcut menu. The **Class Specification** / **Actor Specification** dialog box opens.
2. Click the **Operations** tab. Double-click the desired operation in the tree, or click the **Create** button. The **Operation Specification** dialog box opens.

- Double-click an operation on the Diagram pane or in the Browser.

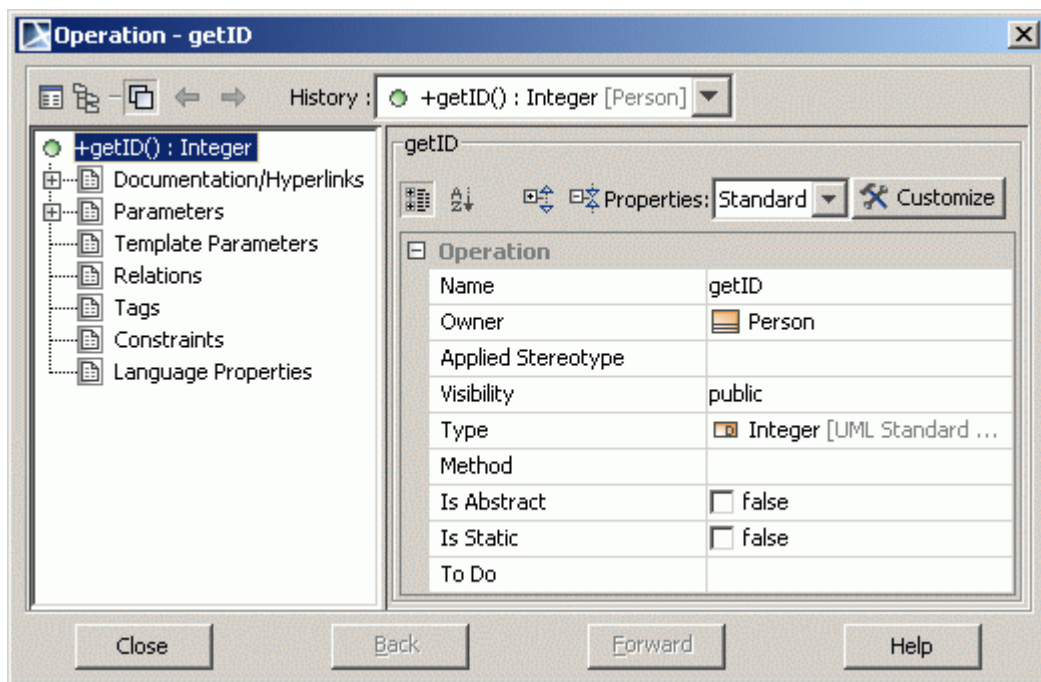


Figure 401 -- Operation Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set a general information for the operation	Owner	Shows a class, which contains the current operation. The value of the Owner field cannot be changed. It is automatically defined when an operation is created.
	Type	Shows an operation type. It can be another project class or primitive type such as int or double. Select the type from the list or create a new one by clicking “...” button.
	Method	Click the “...” button to open the Select Elements dialog box. The Activity, Interaction, or State Machine elements can be assigned from the model.

Tab name	Box	Function
Parameters A parameter is an unbound variable that can be changed, passed, or returned.	Name	Shows the parameter name.
	Type	Shows the parameter type. It can be a classifier or a data type.
	Default Value	An expression whose evaluation yields a value to be used when no argument (operation) is supplied for the parameter.
	Direction	Specifies what kind of parameter is required: <ul style="list-style-type: none"> • return - return parameter. • in - an input parameter (may not be modified). • out - an output parameter (can be modified to communicate information to the caller). • inOut - an input parameter that can be modified.
	Up	Move the list up.
	Down	Move the list down.
	Create	The Parameter Specification dialog box opens.
	Delete	Removes the parameter.

To change an operation name

1. Click the operation in the selected class on the diagram pane or in the Browser tree.
2. Type a new name.
 - Change an operation name in the **Operation Specification** dialog box.

To define the type of an operation

- In the **Operation Specification** dialog box, the **Type** drop-down list box, select the operation type.
- Type a colon ":" and the name of the operation type just after the operation name on the diagram pane. If you specify a nonexistent type of an operation, a new class is created.

To edit / add an operation parameter

1. Open the **Operation Specification** dialog box.
2. Click the **Parameters** tab.
3. Double click on the existing parameter name in the expanded tree or click the **Create** button. The **Parameter Specification** dialog box opens.
 - Type a parameter text (in parenthesis) directly on a diagram.
 - Select an operation in the Browser tree, select **New** from its shortcut menu and select **Parameter**. The **Parameter Specification** dialog box opens.

The Parameter Specification dialog box

The **Parameter Specification** dialog box defines an operation argument.

To open the **Parameter Specification** dialog box

1. Open the **Operation Specification** dialog box.

2. Click the **Parameters** tab, expand tree, and then double-click the desired parameter. Or click the **Create** button.

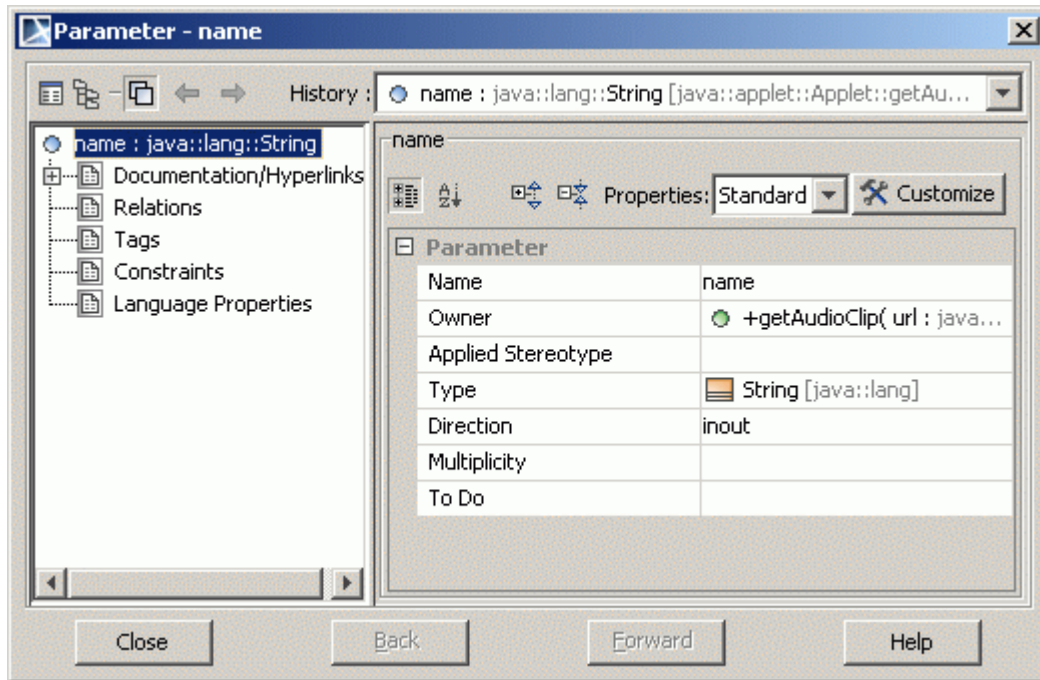


Figure 402 -- Parameter Specification dialog box

Click the **Show Expert Properties** button for showing more properties of the parameter.

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set general information about the parameter	Owner	The name of the operation, which contains the parameter.
	Type	Shows the parameter type. It can be a classifier or a data type. Select a type from the list or create a new one by clicking the “...” button.

Tab name	Box	Function
	Type Modifier	Additional information about the type. <ul style="list-style-type: none"> • & - a parameter is a reference to other model element. • * - a parameter is a pointer to other model element. • [] - a parameter is a an array of other model elements.
	Default Value	An expression whose evaluation yields a value to be used when no argument (operation) is supplied for the parameter.
	Direction	Select the direction kind: <ul style="list-style-type: none"> • return - return parameter. • in - an input parameter (may not be modified). • out - an output parameter (can be modified to communicate information to the caller). • InOut - an input parameter that can be modified.

To add additional information about the return type of an operation

1. Open the **Operation Specification** dialog box.
2. Click the **Show Expert Properties** button. More properties for the operation show.
3. Select a sign from the **Type Modifier** drop-down list box:
 - **&** - one class has a reference to other model element.
 - ***** - one class has a pointer to other model element.
 - **[]** - one class has an array of other model elements.

An operation can be defined as:

Name	Function
Is Abstract	The operation does not have an implementation, and one must be supplied by a descendant.
Is Static	This operation scope means that the values returned by the parameter have no duplicates.
Is Query	The operation does not change the state of the system.

To define an operation as abstract, static, or query

1. Open the **Operation Specification** dialog box.
2. Select the **Expert** mode from the **Properties** field. More properties for the operation show.
3. Select the Is **Abstract**, **Is Static**, and/or **Is Query** check box(es) in the **General** tab.

To set the operation visibility

Visibility name	Function
Public '+'	The operation can be accessed by any other object from the outside.
Package '~'	The operation can be accessed by an element from the same package.
Private '-'	The operation can be accessed only from that class.
Protected '#'	The operation can be accessed from the inside of that class and the classes derived from that class.

1. Open the **Operation Specification** dialog box.
2. From the **Visibility** drop-down list box, select **Public**, **Package**, **Private**, or **Protected**.

NOTE The operation visibility is shown in the operation signature.

To set an operation Concurrency: sequential, guarded or concurrent

Name	Function
Sequential	The callers must coordinate, so that only one call to an Instance (on any sequential Operation) is made at a time. If simultaneous calls occur, then the semantics and the integrity of the system can not be guaranteed.
Guarded	Multiple calls from concurrent threads may occur simultaneously to one Instance (on any guarded Operation), but only one is allowed to commence. The others are blocked until the performance of the first Operation is complete. It is the responsibility of the system designer to ensure that deadlocks do not occur due to simultaneous blocks. The Guarded Operations must perform correctly (or block themselves) in case a simultaneous sequential Operation or guarded semantics cannot be claimed.
Concurrent	Multiple calls from concurrent threads may occur simultaneously to one Instance (on any concurrent Operation). All of them may proceed concurrently with correct semantics. The Concurrent Operations must perform correctly in case a simultaneous sequential or the guarded Operation, or concurrent semantics cannot be claimed.

1. Open the **Operation Specification** dialog box.
2. Select the **Expert** mode from the **Properties** field. More properties for the operation show.
3. Select the concurrency type in the **Concurrency** drop-down list box.

Package

A package groups classes and other model elements together. All types of UML model elements can be organized into packages. Each diagram must be owned by one package and the packages themselves can be nested within other packages. Subsystems and models are special kinds of packages.

The packages may have dependency, generalization, realize, containment, and association relationships. These relationships are usually derived from the relationships between the classes inside those packages.

Working with packages

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the selected package in the **Package Specification** dialog box.

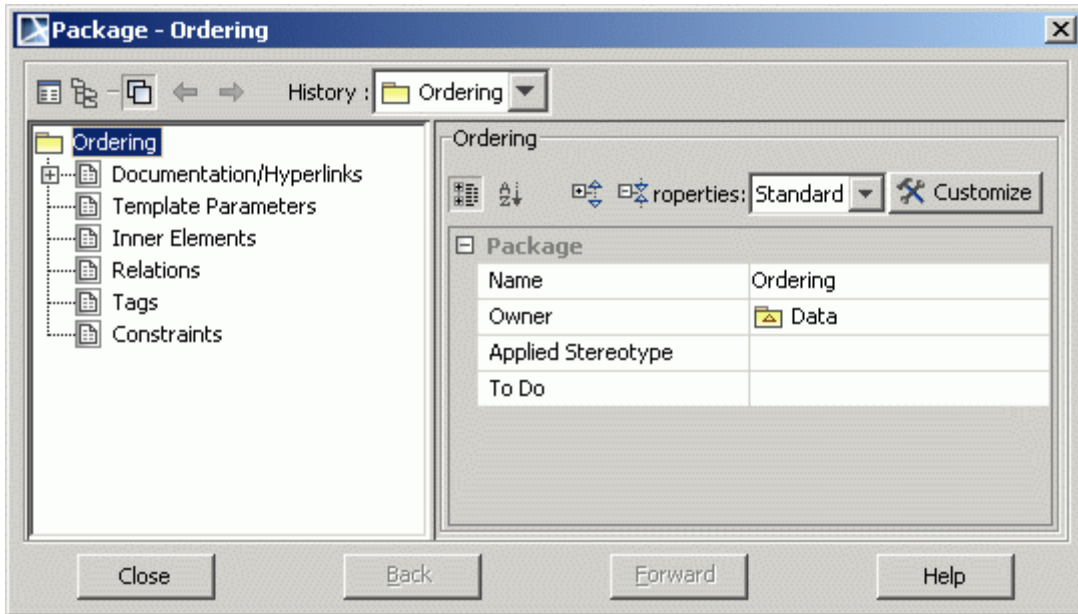


Figure 403 -- Package Specification dialog box

Refer to the "Specification dialog boxes" on page 325 for information about the specification elements not covered in this section.

To add inner elements to the selected package

1. Open the **Package Specification** dialog box.
2. Click the **Inner Elements** tab.
3. Click the **Create** button, and then click an element you wish to add.
4. After selecting the element, the corresponding element **Specification** dialog box opens. If you select a diagram, the **Diagram Specification** dialog box opens.
5. Define the information and click **Back** to return to **Package Specification**.

To change the package header (name, stereotypes, tagged values, and constraints) position

- From the package shortcut menu, select **Header Position** and then **Top** or **In Tab**.
- Select **Symbol(s) Properties** from the package shortcut menu, and then **Header Position Top** or **In Tab** in the **Properties** dialog box.

To show the list of elements assigned to a package on the package shape

Select Show Elements List from the package shortcut menu.



Displaying Package Inner Elements in a Diagram

The **Select Inner Elements** dialog allows you to quickly display inner elements of the package in a diagram.

To display inner elements of the package in a diagram:

1. Right-click a package in a diagram.
2. Select **Related Elements > Display Inner Elements** from the package shortcut menu (Figure 404 on page 958). The **Select Inner Elements** dialog will open (Figure 405 on page 959).
3. Select the elements to be displayed and click **OK**. The selected elements will be displayed in the diagram's package (Figure 406 on page 960).

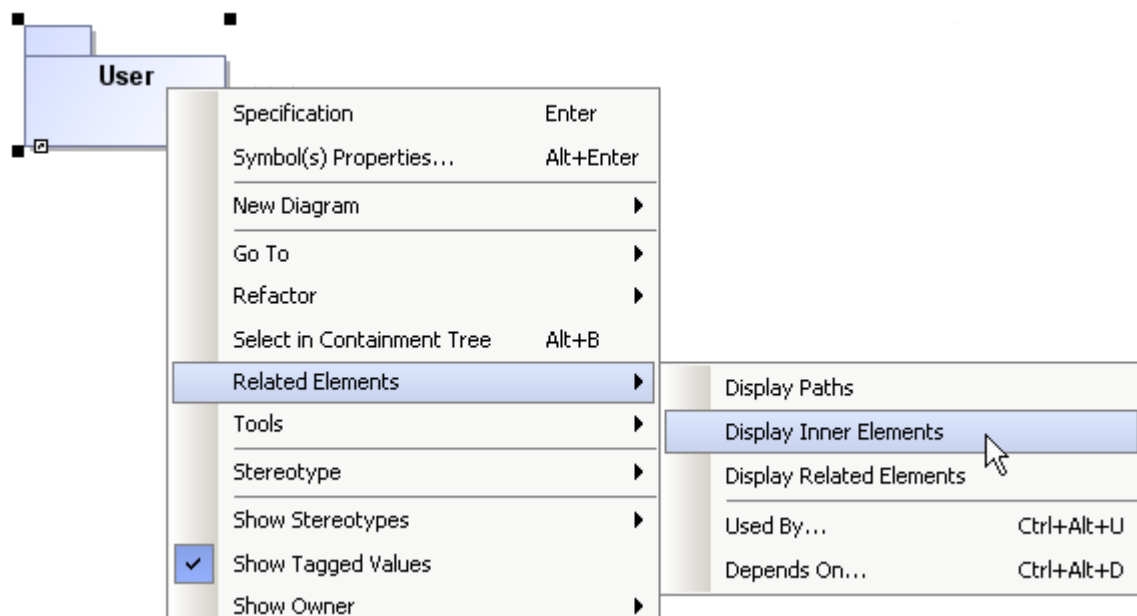


Figure 404 -- Display Inner Elements Menu

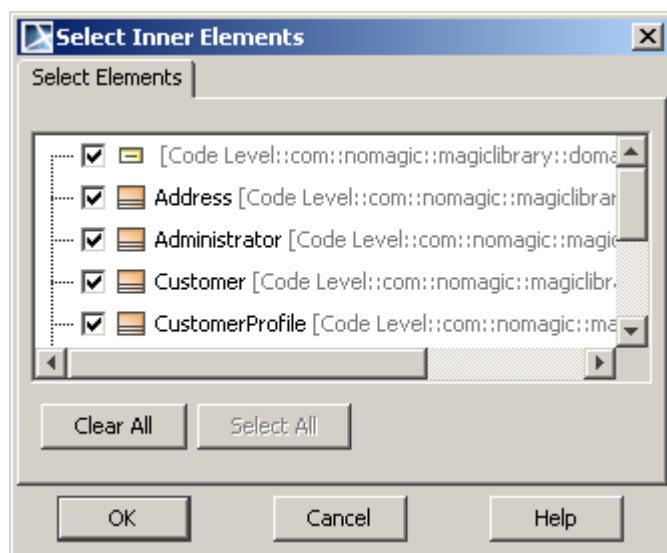


Figure 405 -- The Select Inner Elements Dialog

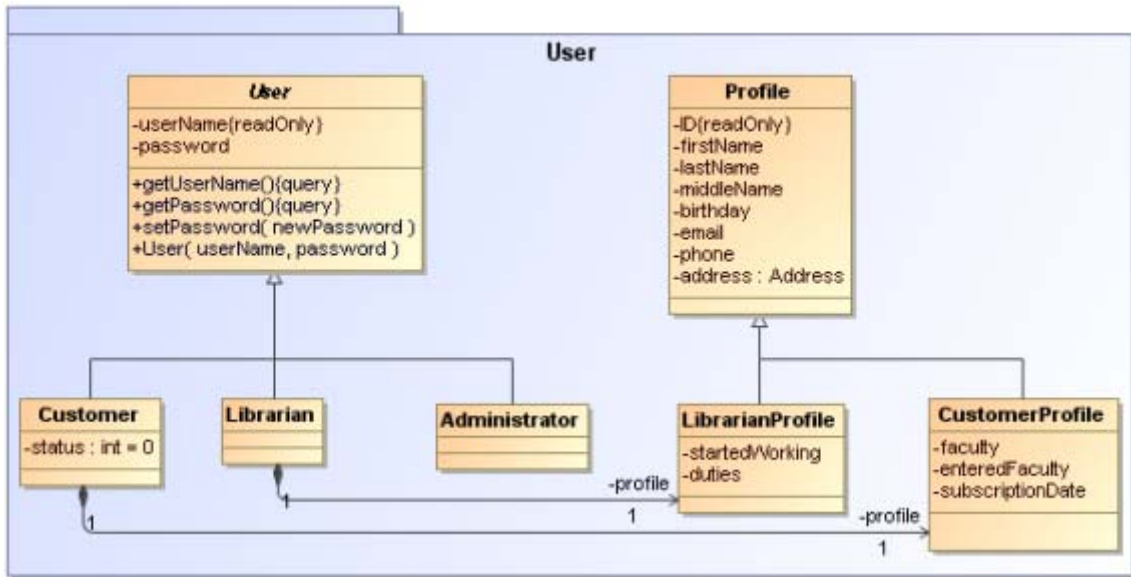


Figure 406 -- User Package with Displayed Inner Elements

Parameter

Parameters synchronization with Arguments

After you have modeled certain references between elements, arguments will be created automatically according to the parameters. Such synchronization increases modeling speed and helps to avoid invalid models.

The table below lists parameter and argument synchronizations.

	Link to synchronization description	Parameter	Argument	Paired elements
1	"Synchronization between Operation parameters and Behavior parameters"	Operation parameter	Behavior parameter	Parameter and paramemeter
2	"Synchronization between Activity parameters and Activity Parameter Nodes"	Activity parameter	Activity Parameter Node	Parameter and activity parameter node
3	"Synchronization between Operation parameters and pins on Call Operation Action"	-Operation parameter -Behavior parameter	- Pin of Call Operation Action - Pin of Call Behavior Action	Parameter and Pin
4	"Synchronization between Interaction parameters and Interaction Use arguments".	Interaction parameter Operation parameter	- Argument of Interaction Use - Argument of Message	Parameter and Argument
5	"Synchronization between Interaction Parameters and Lifelines"	Interaction parameter	Lifeline	Parameter and Lifeline

Created arguments have the same number, order, and name as the parameters. Some properties of the parameters are cloned to argument properties, such as name, type, direction, and multiplicity for a particular argument.

Changes made in parameters are reflected in arguments. Changes to arguments are not reflected in parameters. Exception: synchronization between Activity Parameters and Activity Parameter Node.

When synchronization case is removed, arguments created on synchronization are not removed, but synchronization between parameters and arguments is not working anymore.

For synchronization to work it should match criteria, such as number, order, or other criteria that should be the same for the parameter and argument.

To turn on/off the parameters and arguments synchronization for the whole project, select/clear the **Auto synchronize Parameters and Arguments** check box in the **Project Options** dialog box. After the synchronization is turned off, arguments will not be created and modified on parameters creation and modification. By default the **Auto synchronize Parameters and Arguments** check box is selected.

The new in MagicDraw 15.5 Active Validation functionality improves the parameters and arguments synchronization. Active Validation functionality displays unsynchronized elements on diagram pane and in Browser. You can also use the **Parameters Synchronization** dialog for automatic and manual synchronization solving. For more information about the active validation see "Active Validation", on page 629.

Rules of synchronization between parameters and arguments

In this section you will find information about rules for making the synchronization between parameter and argument function properly. If parameters and arguments do not match rules, they are not synchronized and no changes to arguments will be performed when parameters change.

The general rules of synchronization are:

4. The number of parameters and arguments should be the same.
5. The order of parameters should be the same as order of arguments. If the order of parameters is changed, the order of Arguments is changed too. This is valid if arguments have ordering possibility.
6. The same properties. The parameter properties should be the same as the argument properties. Such as name, type, multiplicity, direction.
7. Properties change. Change the parameter property and the argument property changes. Some of the properties are changed only the first time and after the second change the parameter property is not changed and synchronization is not performed anymore.
8. Each couple of parameters and arguments should be synchronized. If one of them is not synchronized, the other is not synchronized as well. Note that synchronization is checked in element scope.

Synchronization between Operation parameters and Behavior parameters

Term The OMG UML specification (UML 2.2: Superstructure) states:

ownedParameter: Parameter

References a list of parameters to the behavior that describes the order and type of arguments that can be given when the behavior is invoked and of the values that will be returned when the behavior completes its execution.

Definitions *BehavioralFeature* - operation;

Behavior - Activity, Function Behavior, Interaction, Opaque Behavior, Protocol State Machine, State Machine.

Synchronization between operation parameters and behavior parameters works in the following way: after you have assigned a behavior to the operation with parameters, arguments to the behavior will be created automatically.

How synchronization works

After the operation parameters have been modified, arguments change in the following way:

- Create operation parameter - argument is created and properties cloned according to the parameter properties.
- Edit operation parameter - argument properties change. Argument name changes according to parameter properties only on the first parameter name change. For example, for operation create not named parameter. Assign behavior as method to the operation. To behavior not named parameter is created. Now name the operation parameter. Behavior parameter name automatically changes to the operation parameter name. Change the operation parameter the second time. The title of behavior parameter is not changed.
- Remove operation parameter - behavior argument is removed if it does not have links or values. For example, operation parameter is synchronized with activity parameter. Activity parameter is included to other synchronization - activity parameter is synchronized with activity parameter node (see "Synchronization between Activity parameters and Activity Parameter Nodes" on page 967). In this case, after the

operation parameter is removed, the activity parameter is not removed, because it has link.

Validation of Synchronization

Synchronization between Parameter and Argument is valid when:

- The type is compatible. The type of parameter and behavior parameter should be compatible. This means that the type of operation parameter and the type of behavior parameter should be the same or inherited.
- The direction is the same. The direction of the operation parameter and the behavior parameter should be the same.
- The multiplicity is the same. The multiplicity of the operation parameter and the behavior parameter should be the same.

If one of the rules is not valid, operation parameters and behavior parameters will not be synchronized anymore.

Sample

1. Create a class named *Computer*, with an operation called *Collect*, and with parameters *Accepted Computers*, *Production Materials*, and *Rejected Computers*.
2. Create Activity diagram *Collect Computer*.

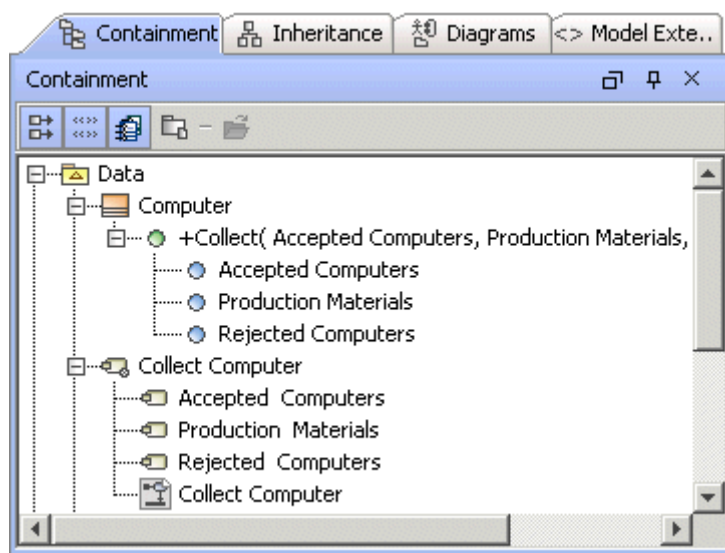


Figure 407 -- Project before synchronization

3. Assign activity diagram *Collect Computer* to the operation *Collect* as behavior. To do this from the operation shortcut menu in the Browser, select the **Behavior Diagram** and then **Assign**. Activity of activity diagram is assigned as a method to the operation

automatically. The text after the name of activity will appear in the Browser: *specifies Collect*.

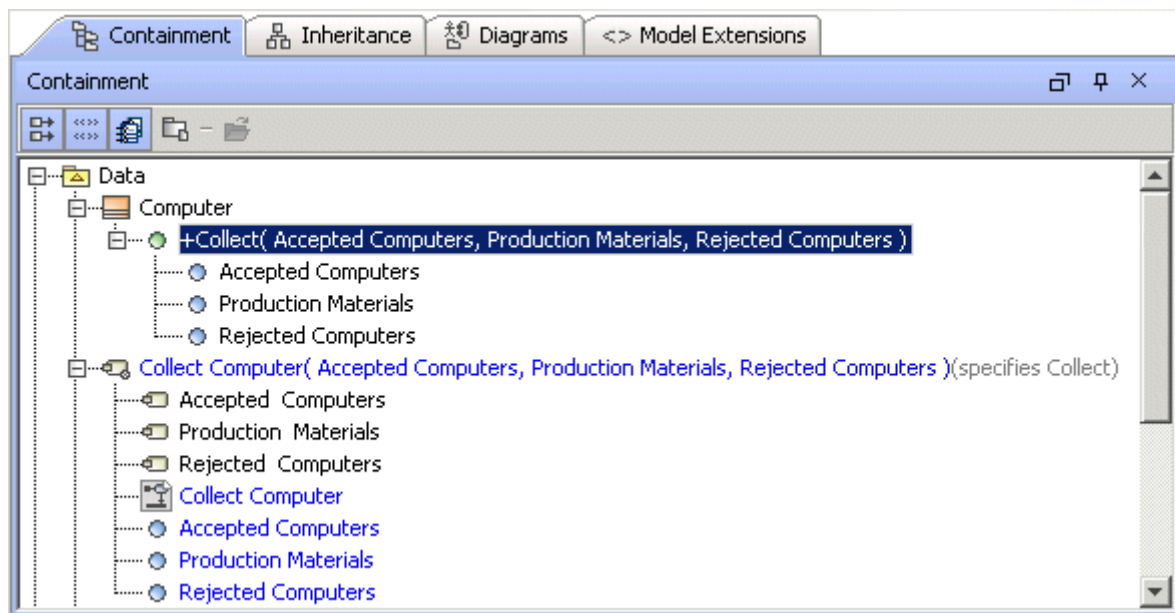


Figure 408 -- Project after synchronization between Operation Parameters and Behavior Parameters

4. The following operation parameters are created to the activity of activity diagram automatically: *Accepted Computers*, *Production Materials*, *Rejected Computers*. The *Collect* operation parameters are synchronized with *Collect Computer* activity parameters.

Synchronization between Activity parameters and Activity Parameter Nodes

Terms The OMG UML specification (UML 2.2: Superstructure) states:

Activity Parameter Node

“Activity parameter nodes are object nodes at the beginning and end of flows that provide a means to accept inputs to an activity and provide outputs from the activity, through the activity parameters.

Activity parameters inherit support for streaming and exceptions from Parameter.”

Constraints

[1] “Activity parameter nodes must have parameters from the containing activity.”

[2] “The type of an activity parameter node is the same as the type of its parameter.”

Definitions *Parameter* - Parameter of Activity;

Argument - Activity Parameter Node.

Synchronization between activity parameters and activity parameter nodes works in the following way: after you have created a parameter to the activity, an activity parameter node of the activity will be created automatically.

How synchronization works ---

How synchronization works on parameter edit:

- Create a parameter in the activity. After you have created the parameter, the argument for parameter will be created automatically. If the parameter direction is *inout* - two activity parameter nodes will be created.
- Remove a parameter from the activity. After you have removed the parameter from the activity, the argument will be removed as well. If two arguments were created for one *inout* parameter, both arguments will be removed.

How synchronization works on argument edit:

- Create an activity parameter node in the activity. The **Select Activity Parameter** dialog box appears. Select the existing activity parameter or click the **Create** button to create a new one.

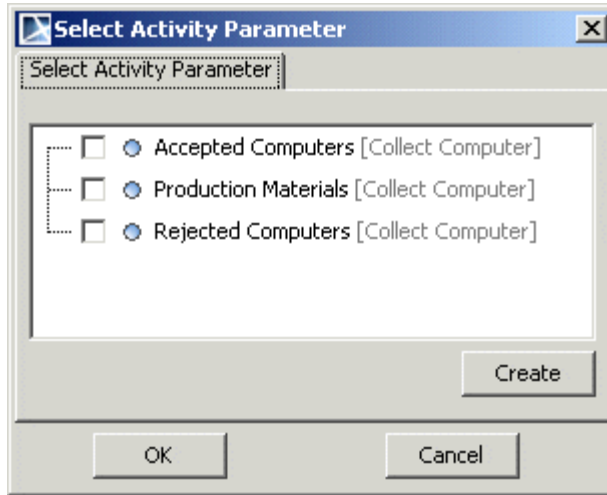


Figure 409 -- Select Activity Parameter dialog box

- Remove synchronized activity parameter node. After you have removed the activity parameter node from the activity, the assigned activity parameter will be removed as well.

NOTES

In the **Select Activity Parameters** dialog box, the following activity parameters are not displayed:

- Inherited parameters
- Parameters which already have Activity Parameter Nodes.

Sample

1. In the *Collect Computer* activity create *Accepted Computers* parameter.
2. The *Accepted Computers* activity parameter node is created in activity automatically. Properties of the parameter and activity parameter node are synchronized.

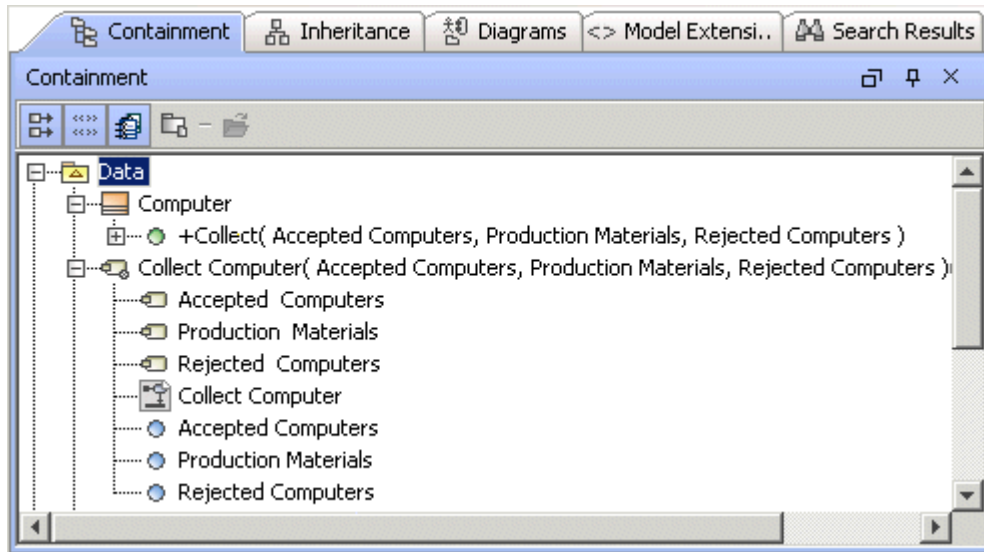


Figure 410 -- Synchronization of Activity Parameters with Activity Parameter Nodes

Synchronization between Operation parameters and pins on Call Operation Action

And

Synchronization between Behavior parameters and pins of Call Behavior Action

Terms The OMG UML specification (UML 2.1.2: Superstructure, p. 244) states:

Call Action

“Constraints

[1] Only synchronous call actions can have result pins.

[2] The number and order of argument pins must be the same as the number and order of parameters of the invoked behavior or behavioral feature. Pins are matched to parameters by order.

[3] The type, ordering, and multiplicity of an argument pin must be the same as the corresponding parameter of the behavior or behavioral feature.”

The OMG UML specification (UML 2.1.2: Superstructure, p. 244) states:

Call Behavior Action

“Constraints

[1] The number of argument pins and the number of parameters of the behavior of type in and in-out must be equal.

[2] The number of result pins and the number of parameters of the behavior of type return, out, and in-out must be equal.

[3] The type, ordering, and multiplicity of an argument or result pin is derived from the corresponding parameter of the behavior.”

Definitions *Parameter* - Parameter of Operation, Parameter of Behavior;

Argument - Pin of Call Operation Action; Pin of Call Behavior Action.

How synchronization works

How synchronization works on parameter edit:

- Create parameter. After you have created a parameter, pins to the Call Operation Action or to the Call Behavior Action will be created automatically. If parameter direction is *inout*, two Pins to one parameter will be created.
- Remove parameter. After parameter is removed, the argument is removed automatically if it does not have links or values.

How synchronization works on argument edit:

- Create pin. Create to the Call Operation Action or the Call Behavior Action input or output pin. The **Select Operation Parameter** dialog box appears for the Call Operation Action element and the **Select Behavior Parameter** dialog box appears for the Call Behavior Action.

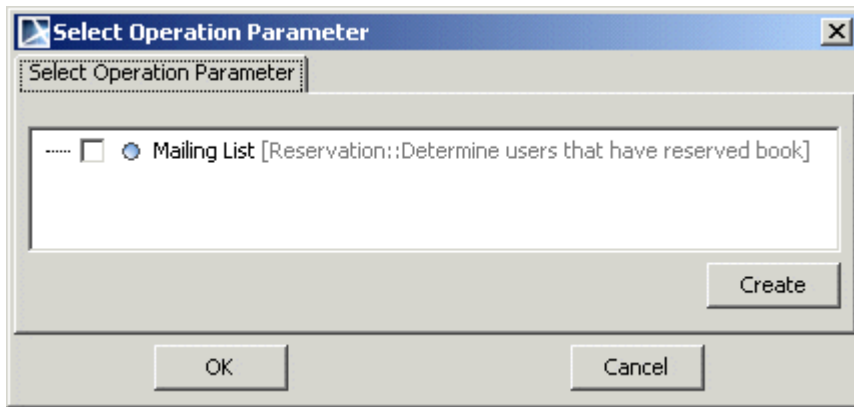


Figure 411 -- Select Operation Parameter dialog box

- Remove pin. After you have removed the pin, the parameter will not be removed.
- The order change and the properties change. After you have changed the order of arguments and after you have changed properties, no changes will be done to the parameters.

Synchronization validation ---

Synchronization between Parameters and Argument is valid if:

- The direction is compatible. The parameter direction and the pin direction should be compatible.

- The types are compatible. The parameter type and the pin type should be the same or inherited.

Sample

1. Create the *Mailing List* parameter to operation.

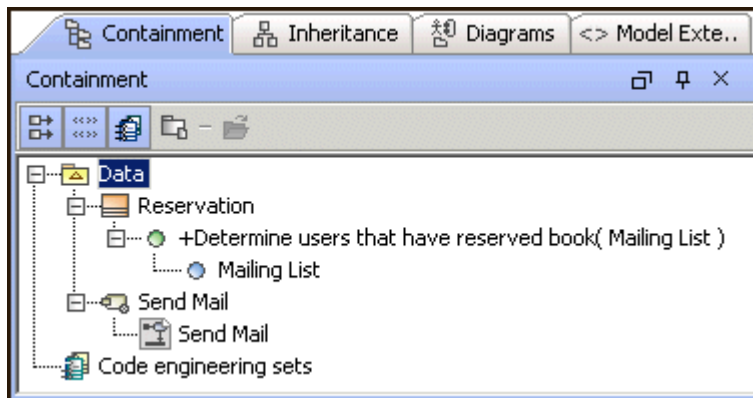


Figure 412 -- Project before synchronization

2. Create the call operation action named *Determine users that have reserved book*.

3. Assign an operation to the call operation action. Pins to the call operation action will be created automatically.

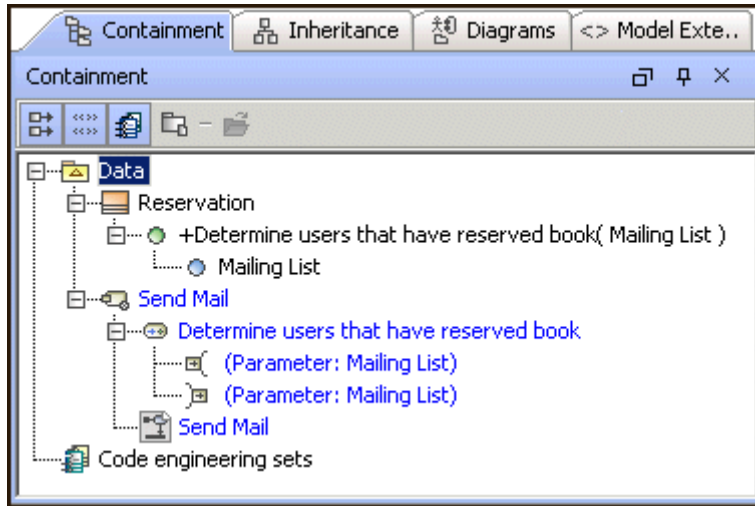


Figure 413 -- Project after synchronization between Operation Parameter and Pin of Call Operation Action

Synchronization between Interaction parameters and Interaction Use arguments

And

Synchronization between Operation parameters and Message arguments

Definitions *Parameter* - Interaction Parameter, Operation Parameters;

Argument - Interaction Use Argument, Message Arguments.

Synchronization between interaction parameters and interaction use arguments works in the following way: after you have referred an interaction use to the interaction with parameter, an argument to the interaction use will be created automatically. The parameter of interaction is synchronized with interaction use argument.

Synchronization between operation parameter and message arguments works in the following way: after you have created the message with an assigned operation, the arguments to the message will be created automatically.

How synchronization works

How synchronization works on parameter edit:

- Create. After you have created interaction use or message, arguments will be created automatically according to parameters.
- Remove parameter. After you have removed the parameter, the synchronized argument will be removed as well.
- The order change. After you have changed the parameters order, arguments order will be changed automatically.
- The property change. After you have changed parameter properties, this change will not affect the argument.

Important Changes in arguments are not reflected in parameters.

Sample

1. Create a class with *Collect* operation and with *Production Material* parameter. Create the sequence diagram. Draw a lifeline with assigned type - *Computer* class. To the lifeline, draw a call message.

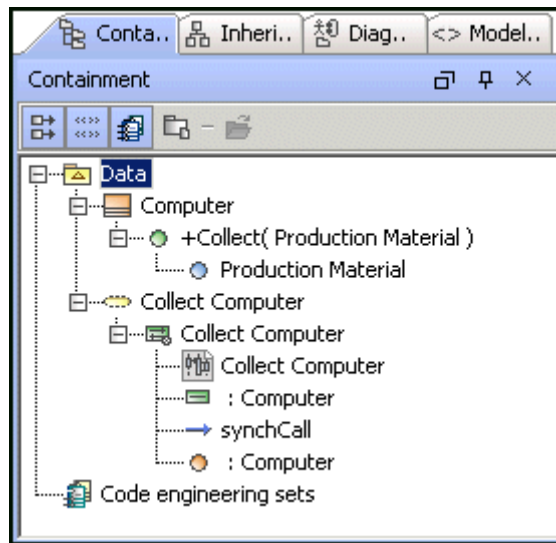


Figure 414 -- Project before synchronization

2. Assign the *Collect* operation to the call message. An argument will be added to the message automatically. To see the created argument, open the **Message** specification dialog box and select the **Arguments** branch. In the sequence diagram on the message, you can see the parameter name with argument name (in this example, the argument is not named).

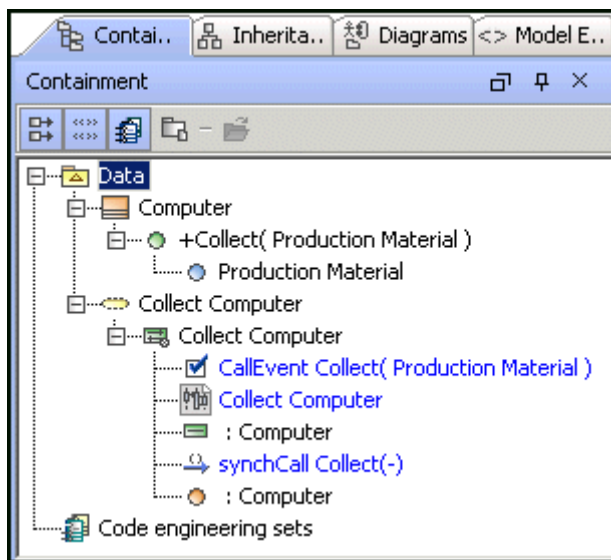


Figure 415 -- Project after synchronization

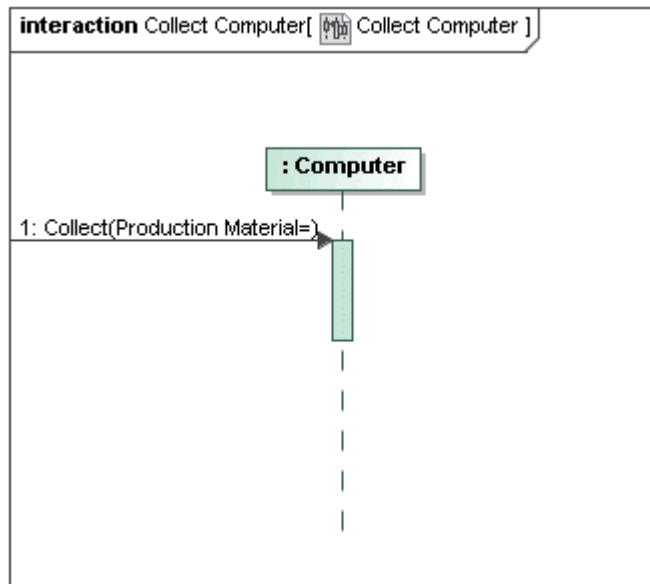


Figure 416 -- Sequence diagram with message with argument

Synchronization between Interaction Parameters and Lifelines

Synchronization between Interaction Parameters and Lifelines works in the following way: after you have created the sequence diagram in interaction with parameters, lifelines for the chosen parameters will be created in the sequence diagram.

This is not the same synchronization as in other cases, because this gives automated lifelines creation from parameters only.

How synchronization works

After you have created, edited, or removed parameters, arguments will be unchanged. Conversely, after you have created, edited, or removed arguments - parameters will be unaffected.

Displaying parameters as lifelines in already existing sequence diagram

You can also display parameters as lifelines in already existing sequence diagram:

1. From the sequence diagram shortcut menu, select **Related Elements** and then select **Display Parameters as Lifelines**. The **Display Parameters as Lifelines** dialog box appears.
2. Select parameters and click **OK**.
3. Lifelines for the selected parameters are created.

Sample

1. Create an interaction with a parameter.

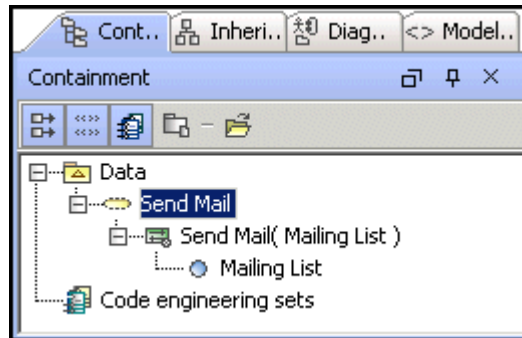


Figure 417 -- Project before synchronization

2. In the Interaction, create the sequence diagram. The **Display Parameters as Lifelines** dialog box appears.

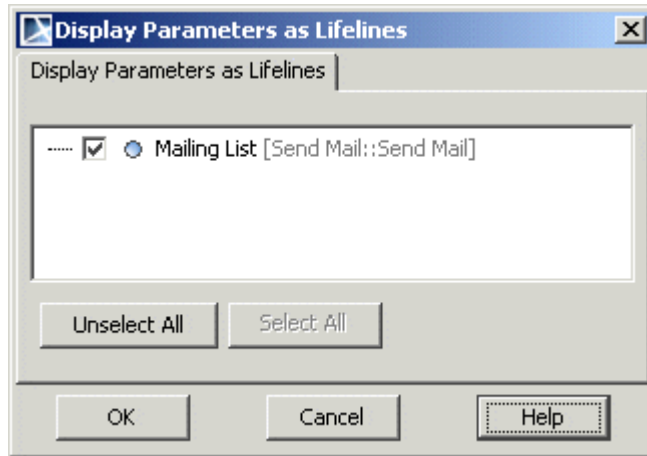


Figure 418 -- The Display Parameters as Lifelines dialog box

3. Select parameters, which will be created as lifelines in the sequence diagram. Click OK.
4. Lifelines are created in the interaction, which is drawn on the sequence diagram.

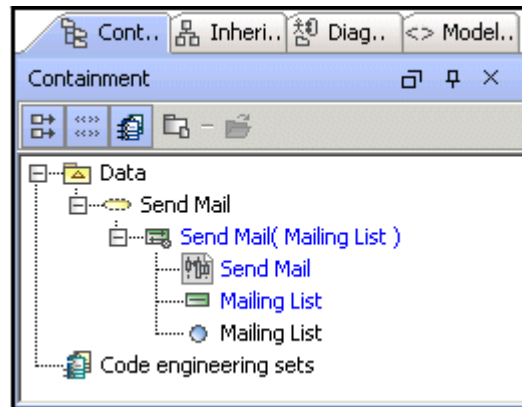


Figure 419 -- Project after synchronization

The Parameters Synchronization dialog box

The **Parameters Synchronization** dialog box (see Figure 420 on page 979) is shown below. It provides useful Parameters and Arguments synchronization options: not synchronized notification, automatic synchronization restoration algorithms, and manual synchronization abilities.

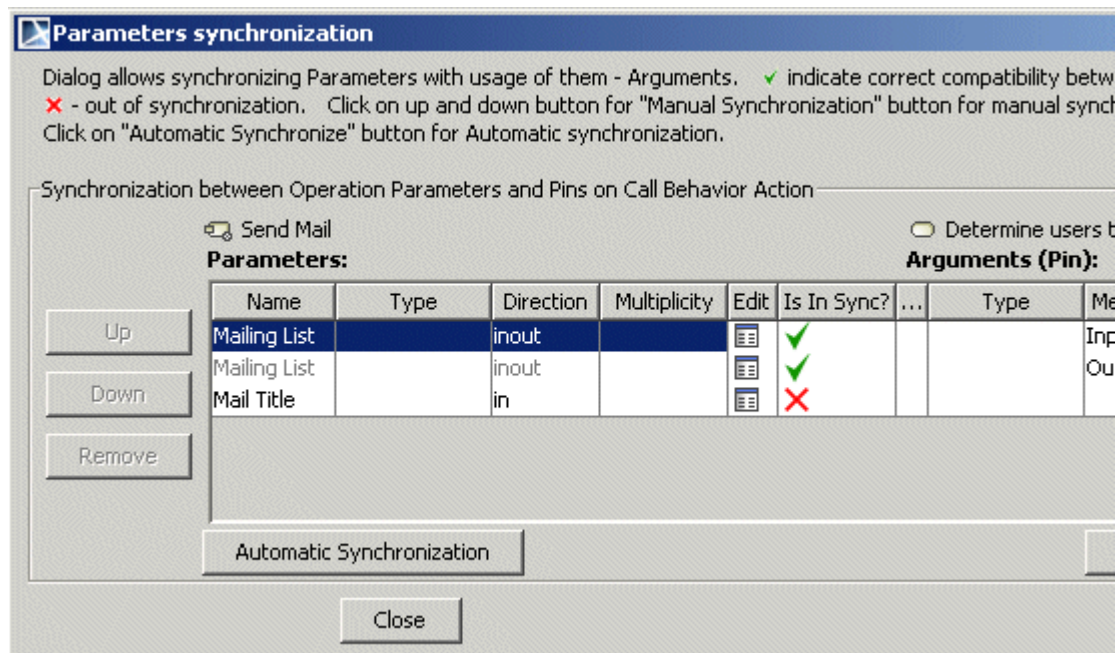


Figure 420 -- The Parameters Synchronization dialog box

There are two ways to open the **Parameter Synchronization** dialog box:

- Select the not synchronized element in the Browser. From the shortcut menu, select the **Validation** > validation group > **Parameter Synchronization** dialog opens.
- Select the invalid element on the diagram. In the symbol smart manipulator, click the invalid element indicator. From the menu that open, select the **Parameter Synchronization dialog** (see Figure 421 on page 980).

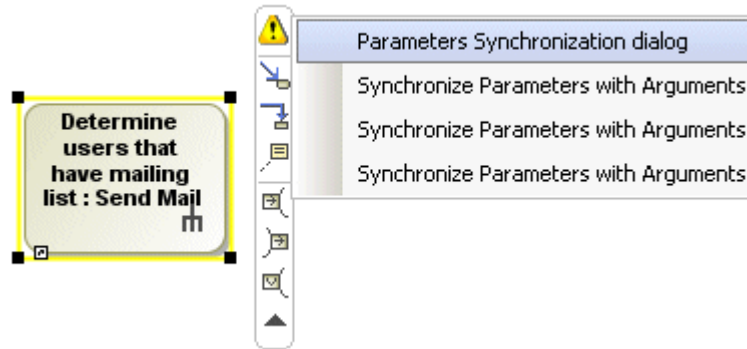



Figure 421 -- Smart Manipulator of the Invalid element

The parameters synchronization dialog box presents the elements those parameters and arguments are not synchronized and provides the possibility to restore the synchronization.

In the **Parameters Synchronization** dialog box, the parameters with arguments are synchronized according to the general synchronization rules, which are described in “Rules of synchronization between parameters and arguments” on page 962.

The **Parameters** group presents information about particular element parameters (see Figure 422 on page 981) and the **Arguments** group presents information about particular element arguments (see Figure 423 on page 982). To edit the parameter or argument, click on the  icon in the **Edit** column (see Figure 424 on page 983). The **Is In Synch?** column displays if parameter and argument are synchronized (see Figure 425 on page 984). Green tick indicates that parameter and argument are synchronized. Red cross indicates that parameter and argument are not synchronized.

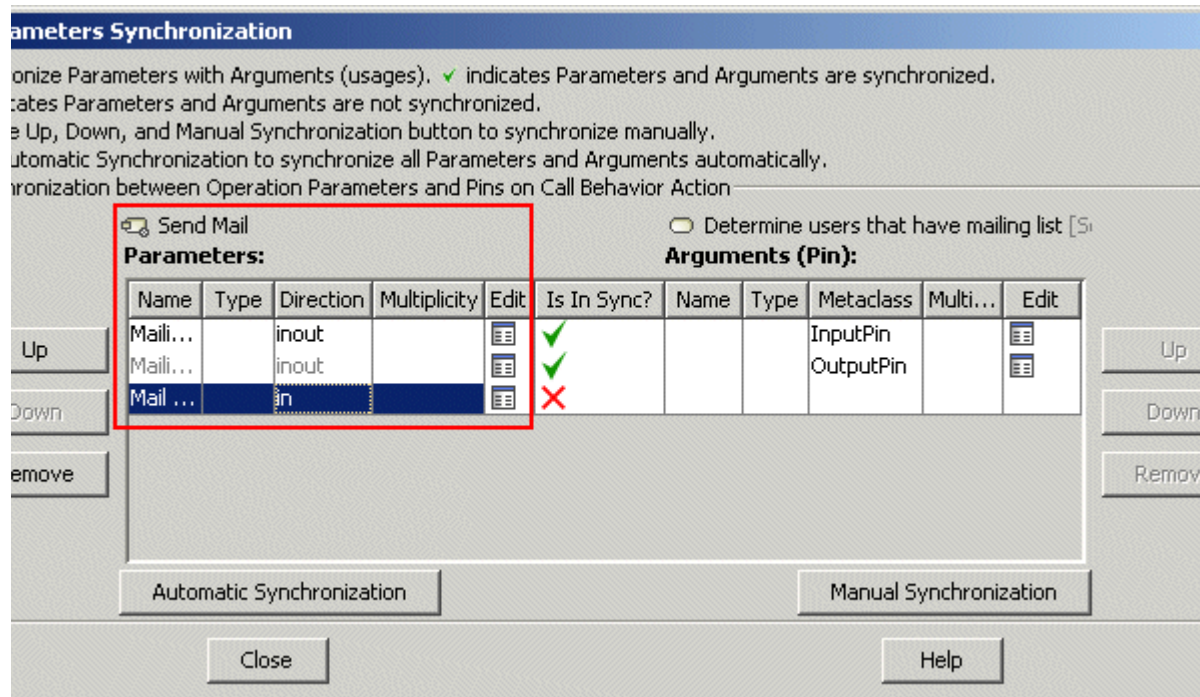


Figure 422 -- The Parameters Synchronization dialog box, Parameters group

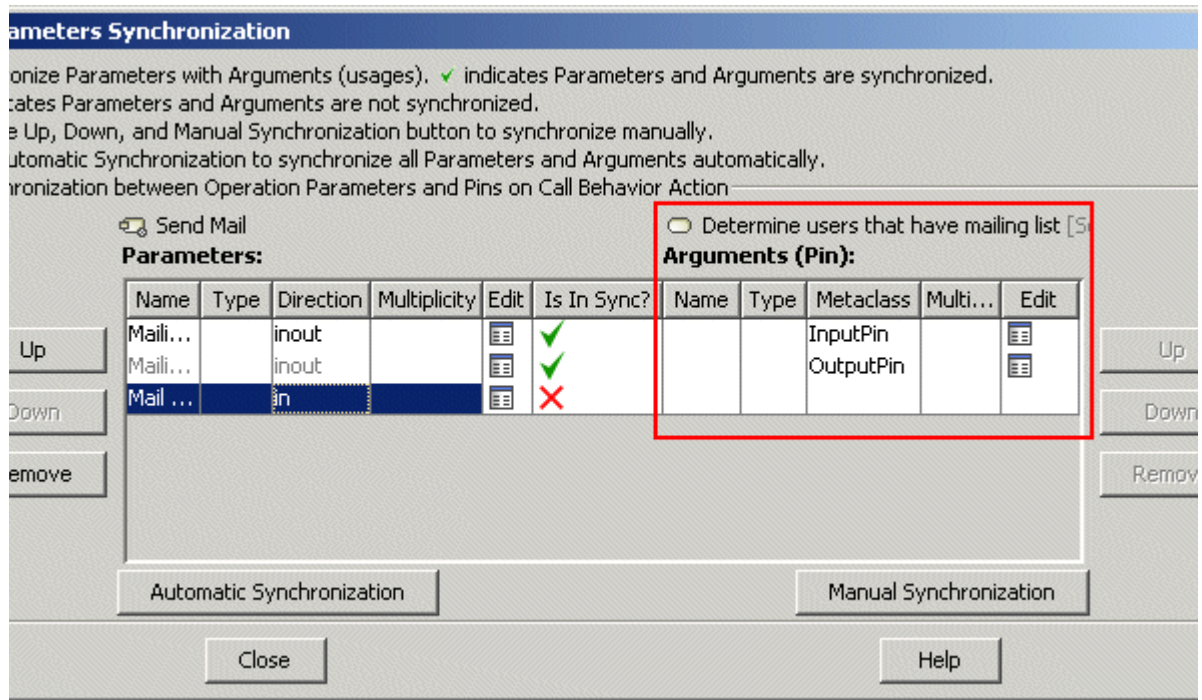


Figure 423 -- The Parameters Synchronization dialog box, Arguments group

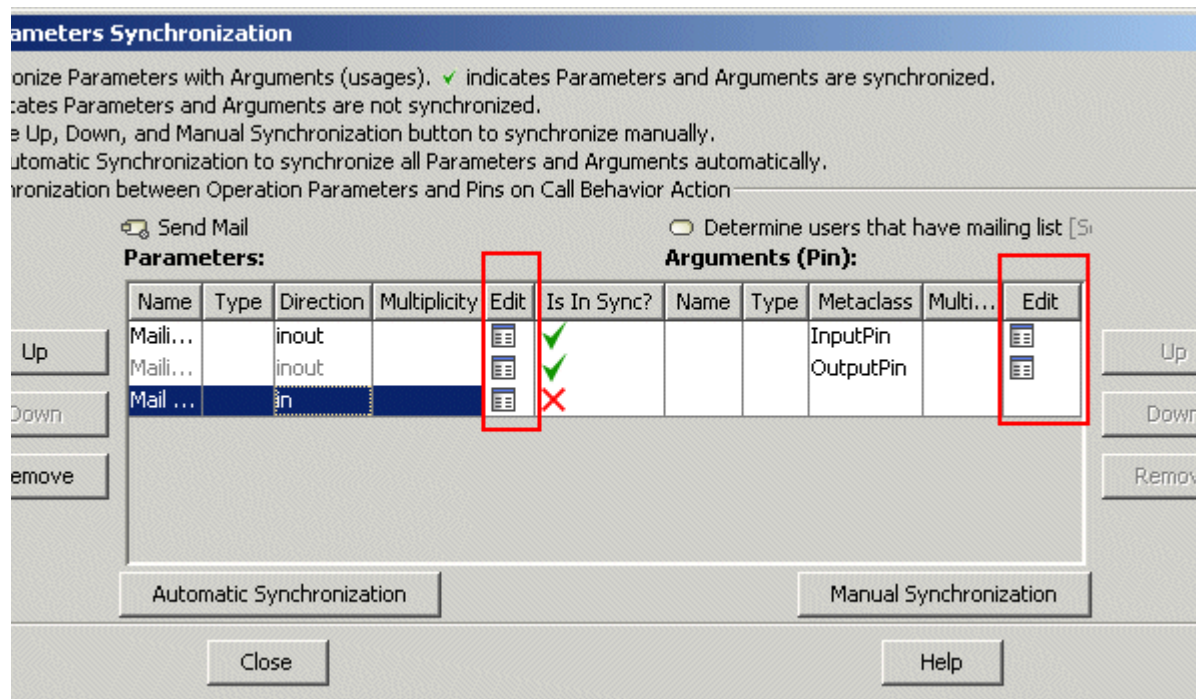


Figure 424 -- The Parameters Synchronization dialog box, Edit column

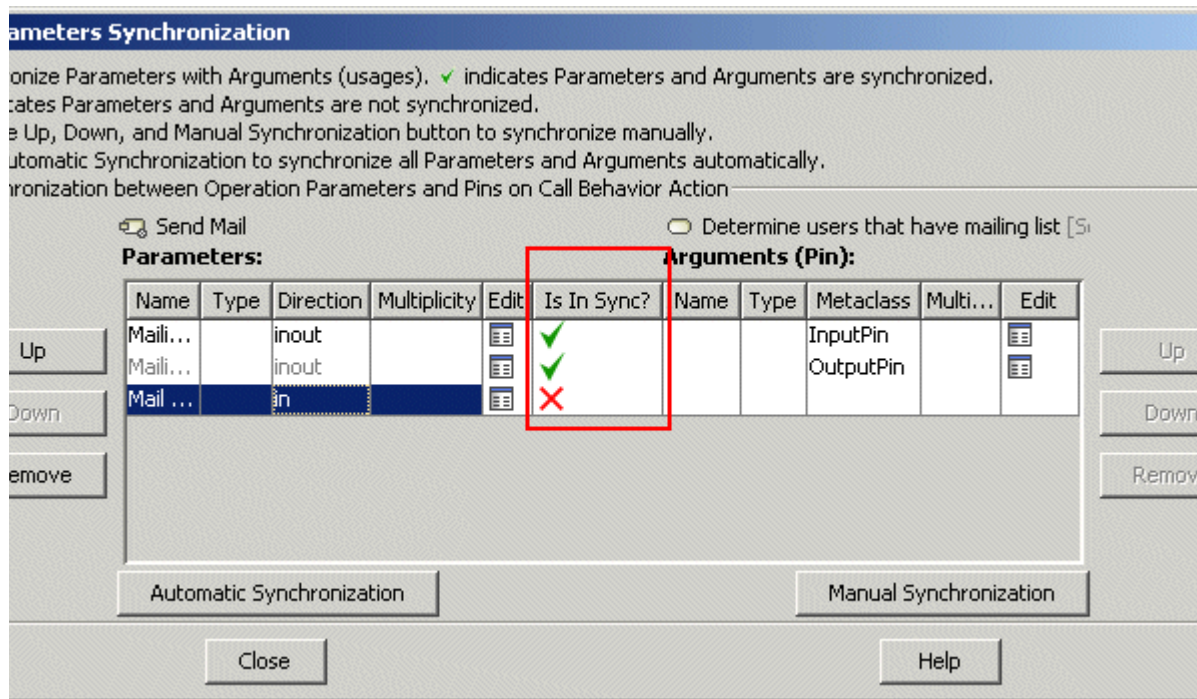


Figure 425 -- The Parameters Synchronization dialog box, Is In Sync? column

Automatic Synchronization

To synchronize parameters with arguments automatically, in the **Parameters Synchronization** dialog box, click the **Automatic Synchronization** button (see Figure 426 on page 985). According to the available synchronization the following available commands appears:

1. Synchronize Parameters with Arguments by restoring initial order and creating missing ones.
 - Missing arguments will be created.
 - Not synchronized Arguments order will be changed and not synchronized properties will be changed according reference.
2. Synchronize Parameters with Arguments by reordering and creating missing ones.
 - Not synchronized arguments order will be changed to fit properties.

- Missing arguments will be created automatically.
 - If reference between parameters and arguments existed, but they are not synchronized or reference will be removed.
3. Synchronize Parameters with Arguments by updating and creating missing ones.
 - Not synchronized arguments not synchronized properties according to parameter will be changed and missing arguments will be created.
 4. Synchronize Parameters with Arguments by creating missing ones.
 - Missing arguments will be created.

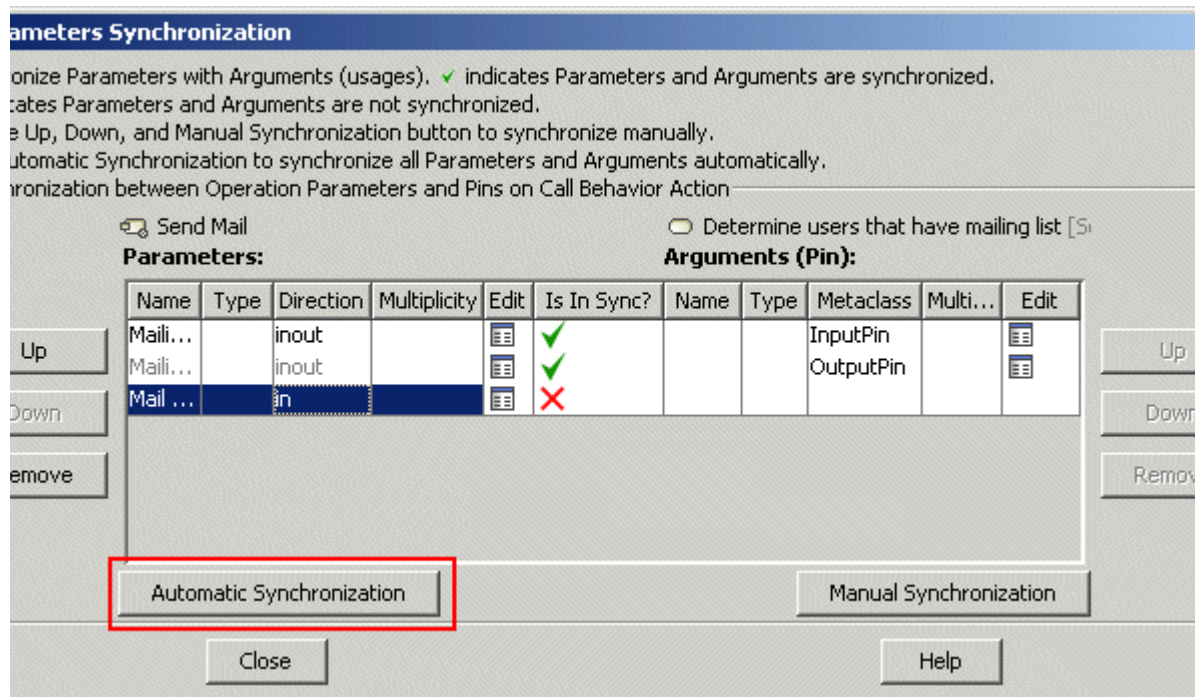


Figure 426 -- The Parameters Synchronization dialog box, Automatic Synchronization

Manual Synchronization

You can manually synchronize parameters with arguments using the **Up/Down/Remove** buttons or click the **Manual Synchronization** button (see Figure 427 on page 987).

Click the **Up/Down/Remove** buttons to move up/down or remove parameters or arguments.

In the **Parameters Synchronization** dialog box table select not synchronized parameter/ argumet and click the **Manual Synchronization** butoon to synchronize manually. See the description of the manual synchronization commands in the table bellow.

Command	Description
<-Clone	Select the <-Clone command to create parameter with the same properties.
Clone->	Select the Clone-> command to create parameter with the same properties.
<-Update	Select the <-Update command to update parameter not synchronized properties in order to be synchronized. The Update item is available if argument exist and there are properties to update.
Update->	Select the <-Update command to update argument not synchronized properties in order to be synchronized. The Update item is available if parameter exist and there are properties to update.

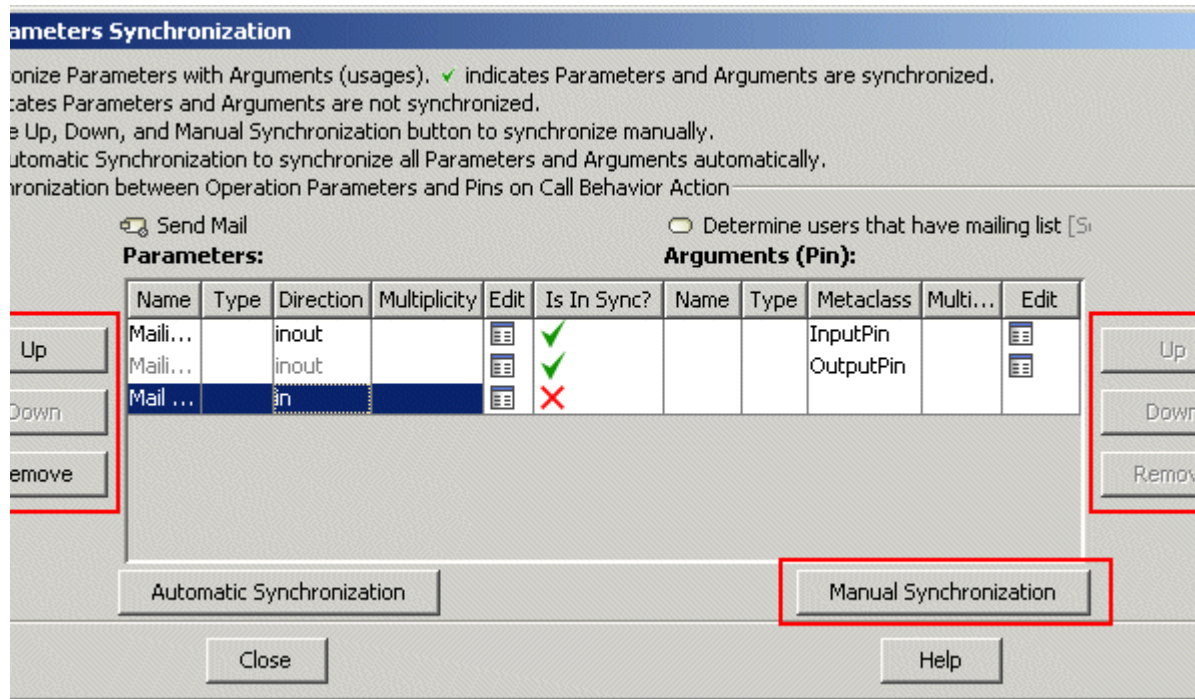


Figure 427 -- The Parameters Synchronization dialog box, Manual Synchronization

Port

A port is a property of a classifier that specifies a distinct interaction point between that classifier and its environment, or between the (behavior of the) classifier and its internal parts. Ports are connected to the properties of the classifier by connectors through which requests can be made to invoke the behavioral features of the classifier.

A Port may specify the services a classifier provides (offers) to its environment as well as the services that a classifier expects (requires) from its environment. It has the ability to specify that any requests arriving at this port are handled.

The Class model element and Component model elements may have any number of Ports.

Define a port in the **Port Specification** dialog box.



Figure 428 -- Port Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box	Function
General Set a general information for the port.	Owner	The name of a class or component, which contains the port.
	Is Behavior	Specifies whether requests arriving at this port are sent to the classifier behavior of this classifier. Such a port are referred to as a behavior port. Any invocation of a behavioral feature targeted at a behavior port will be handled by the instance of the owning classifier itself, rather than by any instances this classifier may contain.

Tab name	Box	Function
	Is Service	If the value of the Is Service check box is true, it indicates that this port is used to provide the published functionality of a classifier. If the value of the Is Service check box is false, it indicates that this port is used to implement the classifier but is not part of the essential externally.
Provided/Required Interfaces	Name	Shows the name of the provided/required interface.
	Type	Shows the type of the interface that specifies a port.
	Add	Drop down menu opens. Select the <i>Provided</i> or <i>Required</i> item and select an interface in the open Select Interface dialog box.
	Remove	Removes an item from the Provided/Required Interfaces list.

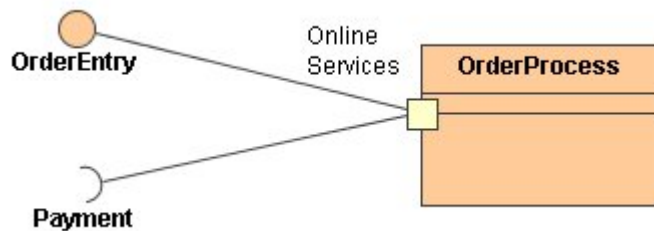
To customize ports list

The ports list can be displayed in a separate compartment on the element shape when the **Suppress Ports** check box is cleared from the element shortcut menu, **Presentation Options** submenu.

1. From the element shortcut menu, select **Edit Compartment** and then **Ports**.
2. In the open **Compartment Edit** dialog box, **Ports** tab, add the desired ports to display from **All** to the **Selected** list.

To draw a Port

In the Implementation diagram toolbar, select a Port element to draw on a class shape.



Draw a Realize path from a port to an interface to depict the Provided Interface.

Draw Usage path from port to interface to depict Required Interface.

Port Improvements

It is now possible to specify the Provided/Required Interfaces for a Port even if the Port type is not specified.

When you add a Provided or Required Interface to a Port, the **Select Port Type** dialog will open (Figure 429 on page 990) with the following options:

- **Set Provided Interface as Port Type** (available on Provided Interface creation only). The Provided Interface will be suggested as the Port Type.
- **Create "dummy" port type automatically.** Create a dummy port type and relations between the type and interface.
- **Select or create a port type manually.** The **Select Port Type** dialog will open to allow you to select or create a Port.

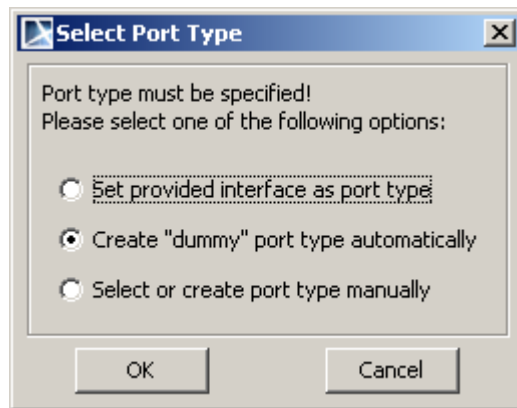


Figure 429 -- The Select Port Type Dialog

Pseudo State

The Pseudo state is typically used to connect multiple transitions into more complex state transitions paths. For example, by combining a transition entering a fork pseudo state with a set of transitions exiting the fork pseudo state, we get a compound transition that leads to a set of orthogonal target states.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Specify the pseudostates in the **Pseudo State Specification** dialog box.

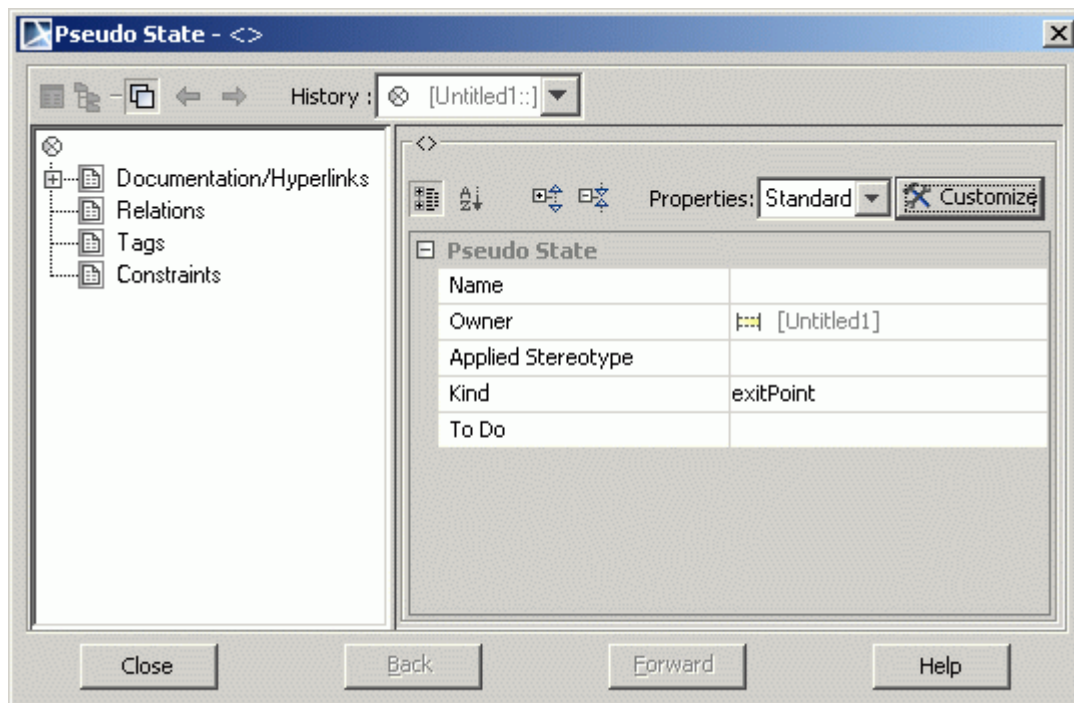


Figure 430 -- Pseudo State Specification dialog box

Tab name	Property	Function
General	Owner	The region, containing a pseudo state.
	Kind	The kind, showing the purpose of the pseudo state.

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Initial

Every object belongs to a particular state as soon as it is created. So, it is useful to explicitly show that particular state. A solid filled circle represents the initial state of an object. There can only be one initial state for an object. The initial state denotes the starting place for a transition, the target of which is a composite state.

Final state

The final state symbol (a circle surrounding a smaller solid circle) is used to represent the object destruction. The final state is optional in the diagram because there is a system that runs without interruption after the start of the activities. Also, there can be several final states in the same state diagram, denoting that the life of the object may finish depending on several conditions.

Terminate

Entering a terminate pseudo state implies that the execution of the state machine by means of its context object is terminated. The state machine does not exit any states nor does it perform any exit actions other than those associated with the transition leading to the terminate pseudo state.

Entry Point

An entry point connection point reference as the target of a transition implies that the target of the transition is the entry point pseudo state as defined in the submachine of the submachine state. As a result, the regions of the submachine state machine are entered at the corresponding entry point pseudo states.

Exit Point

An exit point connection point reference as the source of a transition implies that the source of the transition is the exit point pseudo state as defined in the submachine of the submachine state that has

the exit point connection point defined. When a region of the submachine state machine has reached the corresponding exit points, the submachine state exits at this exit point.

Deep History

The Deep History represents the most recent active configuration of the composite state that directly contains this pseudo state; e.g. the state configuration that was active when the composite state was last exited. A composite state can have at most one deep history vertex.

Shallow History

The Shallow History represents the most recent active substate of its containing state (but not the substates of that substate). A composite state can have at most one shallow history vertex. A transition coming to the shallow history vertex is equivalent to a transition coming to the most recent active substate of a state.

Junction

The junction vertices are semantic-free vertices that are used to chain multiple transitions together. They are used to construct the compound transition paths between states. For example, a junction can be used to combine multiple incoming transitions into a single outgoing transition representing a shared transition path (this is known as merge). Conversely, it can be used to split an incoming transition into multiple outgoing transition segments with different guard conditions.

Choice

The choice vertices, when reached, result in the dynamic evaluation of the guards or the triggers of its outgoing transitions. This realizes a dynamic conditional branch. It allows splitting of transitions into multiple outgoing paths such that the decision on which path to take may be a function of the results of prior actions performed in the same run-to-completion step.

Fork/Join

The fork vertices are used to split an incoming transition into two or more transitions terminating on the orthogonal target vertices (i.e. vertices in different regions of composite state). The segments going out of a fork vertex must not have guards or triggers.

The join vertices are used to merge several transitions emanating from the source vertices in different orthogonal regions. The transitions entering a join vertex cannot have guards or triggers.

Realization

The realization is a specialized abstraction relationship between two sets of model elements, one represents a specification (the supplier) and the other represents an implementation of the latter (the client). The realization can be used to model stepwise refinement, optimizations, transformations, templates, model synthesis, framework composition, etc.

The realization relationship is drawn as a dashed line with a solid triangular arrowhead (a “dashed generalization symbol”). The client (the one at the tail of the arrow) supports at least all of the operations defined in the supplier (the one at the arrowhead), but not necessarily the data structure of the supplier (attributes and associations).

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

The realization paths can be grouped in a tree. This feature makes the appearance of the diagram more structural and understandable.

NOTE

In MagicDraw, you will find three kinds of a realization relationship:

- **Interface Realization.** A dashed line with a solid triangular arrowhead. An Interface Realization is a specialized Realization relationship between a Classifier and an Interface. This relationship signifies that the realizing classifier conforms to the contract specified by the Interface.
- **Realization.** A solid line that represents a relationship between a classifier and an interface.
- **Substitution.** A dashed line with an arrowhead and <<substitute>> stereotype. A substitution is a relationship between two classifiers. It signifies that the substituting classifier complies with the contract specified by the contract classifier. This implies that instances of the substituting classifier are runtime substitutable where instances of the contract classifier are expected.

To create a realization tree if a class or an interface already has a number of realization paths attached to it

Select the **Make Sub Tree** command from the class or the interface shortcut menu.

To remove a realization from the tree or to ungroup a tree

- Select the realization and select the **Remove From Tree** command from the path shortcut menu.
- Select a tree head and select the **Ungroup Tree** command from the tree shortcut menu.

To specify the selected realization path in the **Specification** dialog box

- Double-click the path.
- Select **Specification** from the path shortcut menu.

- Select the path and press ENTER.

Realization and its kinds Specification dialog boxes

The realization, interface realization, and substitution relationships are defined in the dialog box of the same structure. They differ from one another only by the corresponding Specification name..

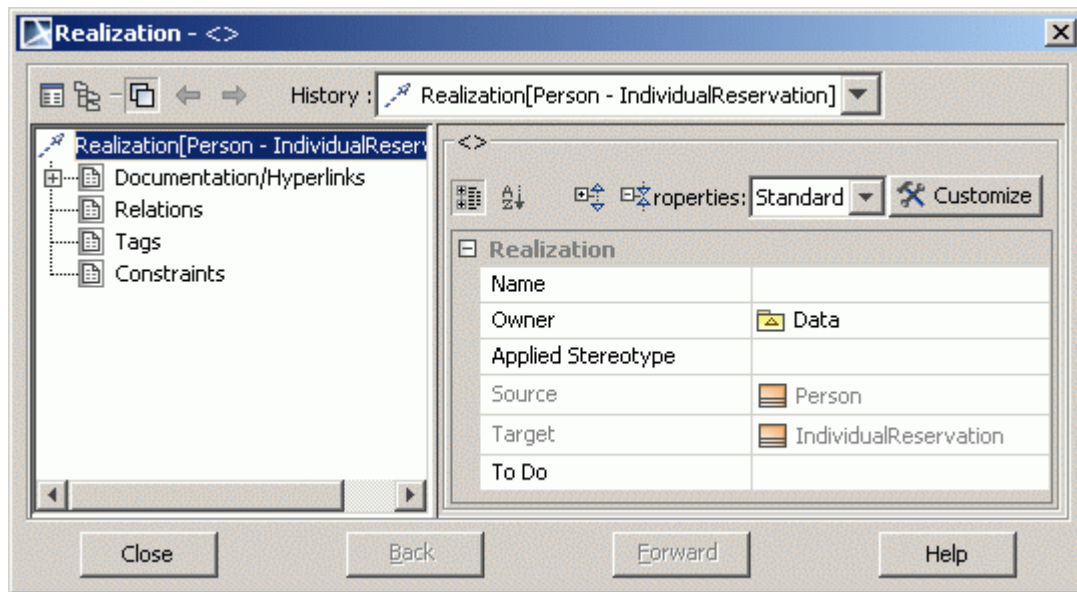


Figure 431 -- Realization Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

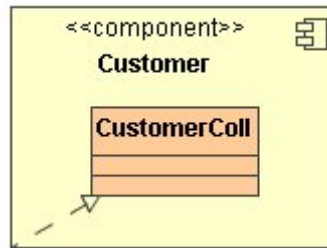
Tab name	Box	Function
General Set general information about the realization relationship	Source	The name of the child element.
	Target	The name of the parent element.
	Mapping	Type a value, or click the ‘...’ button and edit the value in the Edit Mapping dialog box. Note: Available in the Substitution Specification dialog box only.

Creating the realizing classifiers

The realizing classifiers are a set of Realizations owned by the Component. The Realizations reference the Classifiers of which the Component is an abstraction (i.e., that realize its behavior).

To create a Realization relationship between a component and a classifier:

1. Drag the classifier shape to the component shape.
2. Select a classifier or component and select **Related Elements** from its shortcut menu, then select the **Display Paths** command. The realization relationship will be displayed on the diagram pane.



Reception

Term	<p>The OMG UML specification (UML 2.2: Superstructure) states:</p> <p>“A reception is a declaration stating that a classifier is prepared to react to the receipt of a signal. A reception designates a signal and specifies the expected behavioral response. The details of handling a signal are specified by the behavior associated with the reception or the classifier itself.”</p>
-------------	---

Signal receptions can be specified for classes or interfaces.

Parent topic: “Model Elements” on page 796.

Related topics:

“Specification dialog boxes” on page 325.

“Formatting Symbols” on page 342.

“Class” on page 854.

“Interface” on page 917.

To create a new reception

- Double-click the selected class or select **Specification** from the class shortcut menu. The **Class** specification dialog box opens. Click the **Signal Receptions** tab and then click the **Create** button. The **Select Signal** dialog box opens. Select a signal or create a new one. Click **OK**. The **Signal Reception** specification dialog box opens. Specify a new reception and click **OK**.
- Select a class in the Browser tree. From the class item shortcut menu, select **New** and then select **Signal Reception**.
- Select a class shape and click the small red **Insert New Signal Reception** smart manipulation button.

NOTE The signal reception compartment is suppressed and the smart manipulator button is not visible by default.

- Drag-and-drop the signal symbol on the class or interface shape in the diagram pane. The signal reception with the assigned signal is created. You may also drag a signal from the Browser to the class or interface shape on the diagram pane.

To open the Signal Reception specification dialog box

1. Open the **Class** specification, or the **Interface** specification dialog box.
 2. In the **Signal Receptions** tab, double-click the desired signal reception in the tree, or click the **Create** button.
- OR
- Double-click the desired signal reception directly on the diagram.

Refer to “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

To set the signal for a signal reception

- Create a signal reception. The **Select Signal** dialog box opens. Select an existing or create a new signal for the signal reception.
- In the **Signal Reception** specification dialog box, in the **Signal** drop down menu, select a signal. You may also click the "...“ button. In the **Select Element** dialog box, select a signal or click the **Create** button to create a new one.

To display the signal reception on the diagram pane

The signal reception compartment is added to the class and interface shape. This compartment is hidden by default. To show the signal reception compartment:

- From the symbol shortcut menu, select the **Presentation Options** command, then select the **Suppress Signal Receptions** command.
- From the symbol shortcut menu, select the **Symbol(s) Properties** command. In the element **Properties** dialog box, clear the **Suppress Signal Receptions** command.

To show/hide the signal receptions in the Signal Reception compartment

Select the **Edit Compartment** command from the symbol shortcut menu. Select the **Signal Reception command**. The **Edit Compartment** dialog box opens. Only those signal receptions will be displayed in the Signal Reception compartment that are displayed in the **Selected** list.

To change the order of the signal receptions

- From the symbol shortcut menu, select the **Presentation Options** command, then select the **Signal Receptions Sort Order** command.
- From the symbol shortcut menu, select the **Symbol(s) Properties** command. In the element **Properties** dialog box, clear the **Suppress Signal Receptions** check box.

Send Signal Action

The Send Signal Action is an action that creates a signal instance from its inputs and transmits it to the target object, where it may cause the start of the state machine transition or the execution of an activity. The argument values are available to the execution of associated behaviors. The requester

continues the execution immediately. Any reply message is ignored and is not transmitted to the requester.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the selected send signal action in the **Send Signal Action Specification** dialog box.

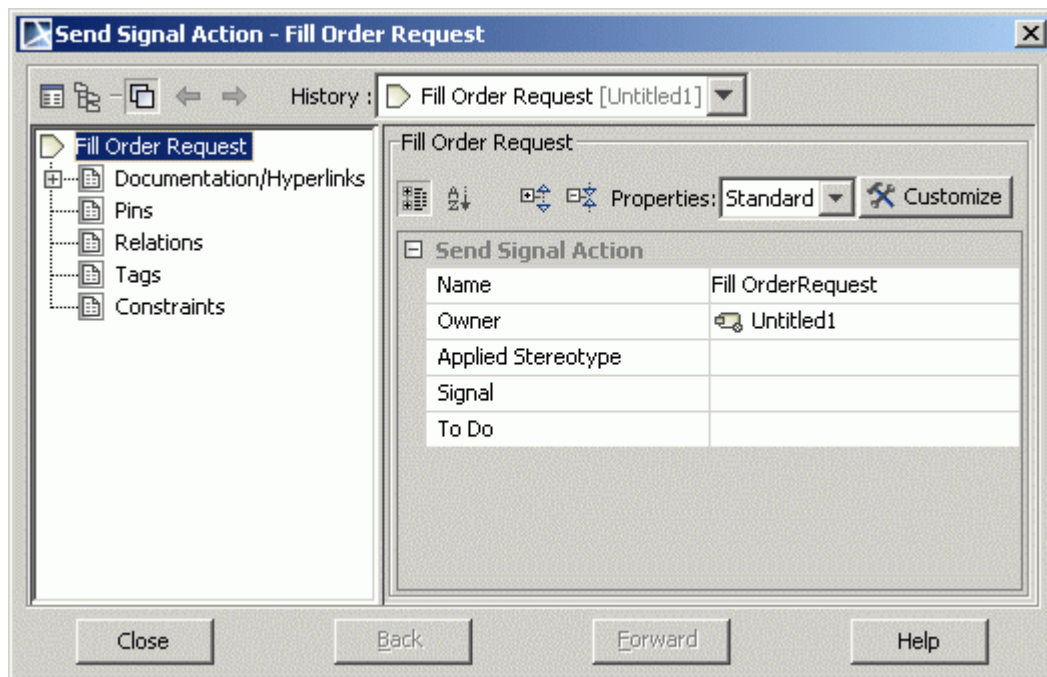


Figure 432 -- Send Signal Action Specification dialog box

Refer to the "Specification dialog boxes" on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General Set a general information about the object node.	Signal	Click the "..." button to assign an existing signal from the model in the Select Elements dialog box, or click Create to assign a new one.

State

A state is a condition during the lifetime of an object or an interaction during which the object meets certain conditions, performs an action, or waits for an event. The state is defined by the concepts of duration and stability. An object may not be in an unknown or undefined state. A state may have two compartments to provide more information about that state:

- The first compartment is the name compartment, it contains the state name, for example: running, going up.
- The second compartment is the activity compartment, it contains the events and actions of the state.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

MagicDraw supports four types of states: State, Composite State, Orthogonal State, and Submachine State. Define the selected state in the **Composite State Specification** dialog box.

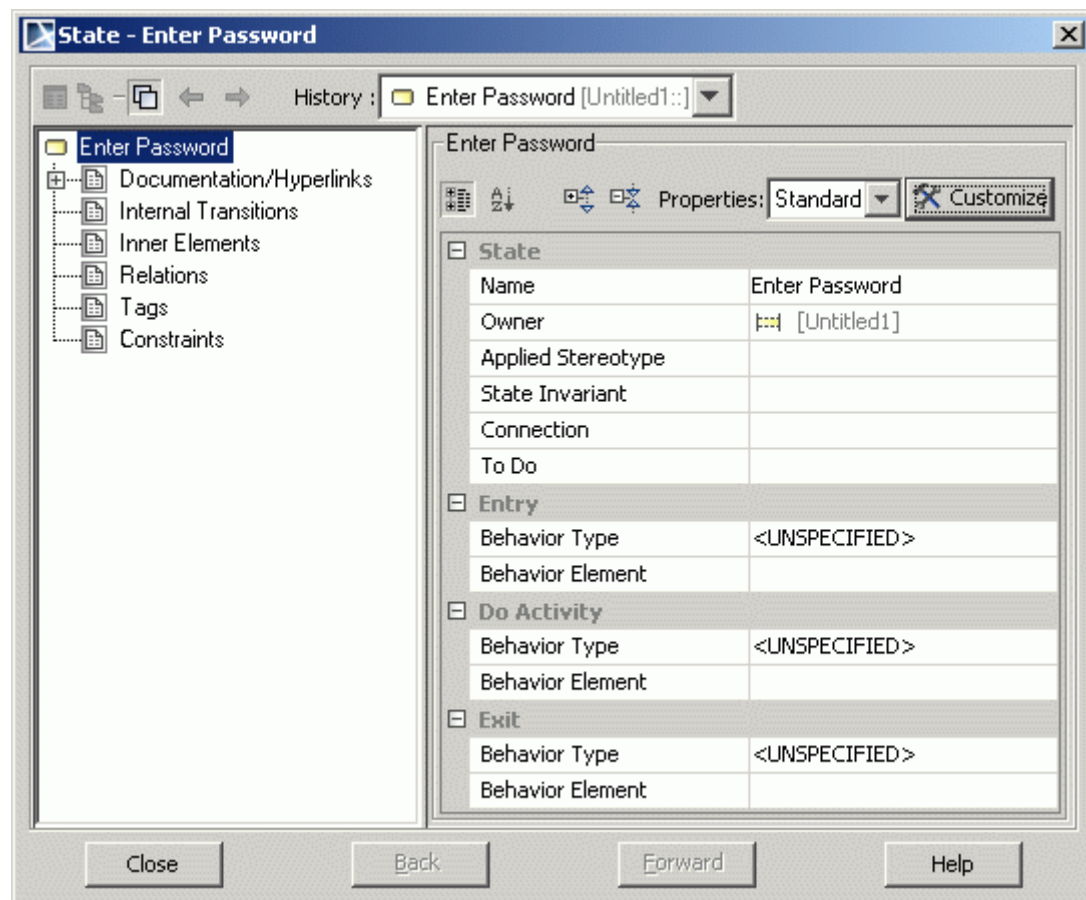


Figure 433 -- State Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General Contains the state actions, events and buttons for editing the list.	State Invariant	Type text or click ‘...’ to open the Edit State Invariant dialog box.
	Connection	Click “+” to open the Connection Point Reference dialog box. Specify the information for the connection.
	Actions group box	
	Entry	
	Behavior Type	Select a type from the activity, interaction, or state machine items list.
	Behavior Element	Click ‘...’ and select an element from the activity, interaction, or state machine items list. The corresponding dialog box opens.
	Do Activity	
	Behavior Type	Select a type from the activity, interaction, or state machine items list.
	Behavior Element	Click ‘...’ and select an element from the activity, interaction, or state machine items list. The corresponding dialog box opens.
	Exit	
	Behavior Type	Select a type from the activity, interaction, or state machine items list.
	Behavior Element	Click ‘...’ and select an element from the activity, interaction, or state machine items list. The corresponding dialog box opens.
Inner States The list of inner states and buttons for editing them	Name	The name of the inner state.
	Type	The type of the inner state.
	Region	The region, to which an inner state was added.

Tab name	Box name	Function
Internal Transitions A set of transitions that, if triggered, occurs without exiting or entering the state.	Name	The name of the internal transition.
	Create	The Transition Specification dialog box opens. Define the transition.
	Delete	Remove the selected transition from the state.

To suppress/unsuppress the actions compartment

In the state shortcut menu, select/clear the **Suppress Actions** check box.

To insert a new region to the state

- In the state shortcut menu, select **Insert New Region**.

To insert a new inner state to the state

- In the Browser, drag and drop the selected state to a Region.
- On the diagram pane, select the state and drag and drop it on the state symbol.

To display region name on the state symbol

Region name can be optionally displayed on the State symbol on a diagram.

To display or hide region name:

- On a diagram from the State symbol shortcut menu check or clear the **Show Region Name** check box.
- In the State **Properties** dialog box select or clear the **Show Region Name** check box.

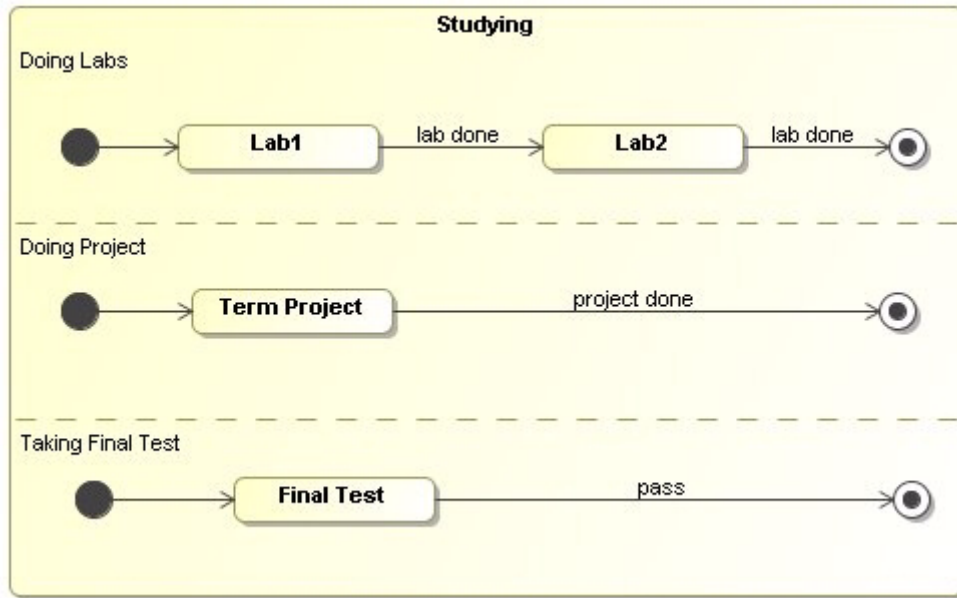


Figure 434 -- Displaying region name on the State symbol

Changing State to Composite/submachine/orthogonal State

You can change your current state to a simple state, composite state, orthogonal state, and submachine state.

To change the state to the composite state

1. From the State diagram toolbar, select the State to draw on the diagram pane.
2. From the state shortcut menu, select the **Add New Region** command. The region is added and the state shape enlarges.

To change the state to the orthogonal state

1. From the State diagram toolbar, select the State to draw on the diagram pane.
2. From the state shortcut menu, select the **Add New Region** command. The region is added and the state shape enlarges.

3. Open the state shortcut menu again and add the second region to the state.

To change the state to the submachine state

1. From the State diagram toolbar, select the State to draw on the diagram pane.
2. From the state shortcut menu, select the **Submachine** command and select an existing State Machine from the list, or click **New** to create a new one.

NOTE : To change the state to the submachine state, all regions from the state should be removed. To do this, select the state on the diagram pane and from the shortcut menu, select the **Remove Region** command.

Composite State

A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions. A given state may only be decomposed in one of these two ways.

Any state enclosed within a region of the composite state is called a substate of that composite state. It is called a direct substate when it is not contained by any other state; otherwise it is referred to as an indirect substate.

Each region of the composite state may have an initial pseudostate and a final state. A transition to the enclosing state represents a transition to the initial pseudostate in each region.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the selected composite state in the **State Specification** dialog box. For a detailed description of this dialog box, see "State Specification dialog box" on page 1002.

For more information on working with states, see "State" on page 1001.

To add a new region to the composite state

Select **Add New Region** from the composite state shortcut menu.

To remove a region from the composite state (at least two regions have to be left)

Select **Remove Region** from the composite state shortcut menu and select a region you want to remove.

Submachine

A submachine state specifies the insertion of the specification of a submachine state machine. The state machine that contains the submachine state is called the containing state machine. The same state machine may be a submachine more than once in the context of a single containing state machine.

The submachine state is semantically equivalent to a composite state. The regions of the submachine state machine are the regions of the composite state. The entry, exit, behavior actions, and internal transitions, are defined as part of the state. The submachine state is a decomposition mechanism that allows factoring of the common behaviors and their reuse.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

Define the selected submachine in the **State Specification** dialog box. For a detailed description of this dialog box, see "State Specification dialog box" on page 1002.

For more information on working with states, see "State" on page 1001.

Adding connection point reference

The connection point reference represent an entry to or exit from the submachine state. It can be used as the source or target of a transition.

To draw the connection point reference on the submachine state

1. In the state diagram toolbar, click the **Connection Point Reference** button. Click on the diagram pane on the submachine state shape. The **Select Entry/Exit Point** dialog box opens.
2. Select *entry point* to define the entry into the submachine state or *exit point* to define the exit from the submachine state. The Connection Point Reference is drawn on the submachine state with a defined entry or exit point.

To see the assigned entry/exit point, open the **Connection Point Reference** dialog box. The **Entry** or **Exit** properties will display the defined entries.

To assign the entry/exit points to the Connection Point Reference

Select the Connection Point Reference on the diagram pane. Open the shortcut menu and select the **Select Entry/Exit Point** command.

NOTE:

In the **Select Entry/Exit Point** dialog box, only these entry and exit points are listed, which are created at the same State Machine as the submachine state. If there are no entry/exit points at the same state machine, the **Select Entry/Exit Point** dialog box is not opened when drawing the Connection Point Reference.

Defining State Invariant

To define a state condition

- Double-click on the state to open the **State Specification** dialog box and in the **State Invariant** field, type an condition and submit changes.
- Near the **State Invariant** field, click "..." button. The **Edit State Invariant** dialog box opens. Type the condition and close the dialog box by submitting changes.

The State Invariant value is displayed on the diagram pane on the state shape in brackets:

Typing password
[State Invariant expression]

Assigning behavior to state

You may define a behavior to be executed correspondingly to the listed events while being in the state whenever the state is entered and exited.

To assign a behavior to a state

1. Double-click the state to open the **State Specification** dialog box.

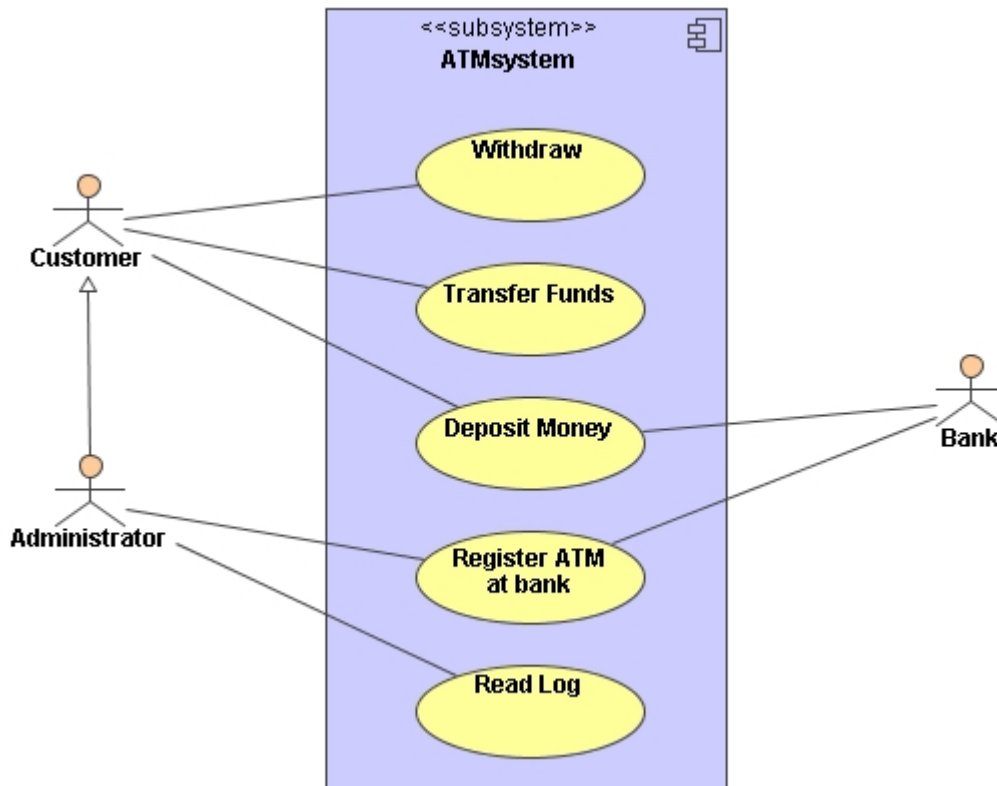
In the Do Activity, Entry or Exit group, the Behavior Type combo box, assign one of the behaviors: Activity, Interaction, or State Machine.

Subsystem

A subsystem is treated as an abstract single unit. It groups model elements by representing the behavioral unit in a physical system.

To draw a subsystem

1. In the Use Case diagram toolbar, select the Subsystem element to draw.



Swimlane

Actions and subactivities can be organized into swimlanes in the activity diagrams. The swimlanes are used to organize responsibility for actions and subactivities according to the class. They often correspond to the organizational units in a business model.

The swimlanes limit and provide a view on the behaviors invoked in the activities. They consist of one or more partitions. They may be vertical and horizontal.

An activity diagram can be divided visually into “swimlanes,” each separated from the neighboring swimlanes by vertical or horizontal solid lines on both sides. Each swimlane represents a responsibility for part of the overall activity, and may eventually be implemented by one or more objects. The relative ordering of the swimlanes has no semantic significance, but may indicate some affinity. Each action is assigned to one swimlane. Transitions may cross lanes. There is no significance to the routing of a transition path.

To open the **Activity Partition Specification** dialog box

Select **Specification** from the partition (swimlane) shortcut menu, or double-click the partition line.

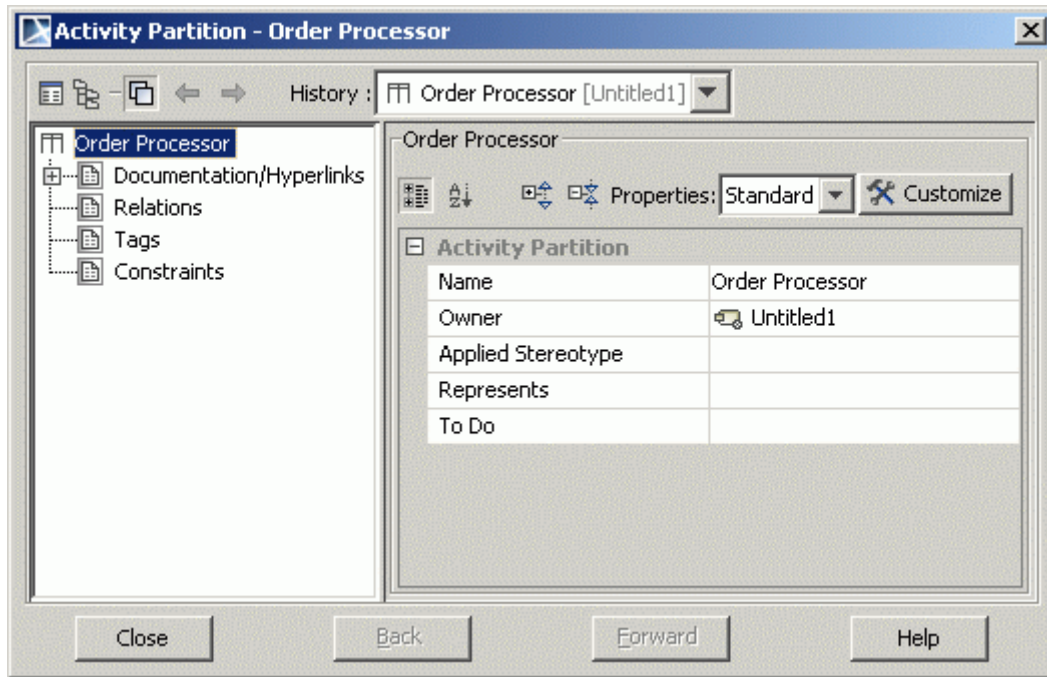


Figure 435 -- Activity Partition Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General Set general information about the partition	Owner	Shows the name of the activity diagram where the partition is placed.
	Represents	Click the “...” button to open the Select Elements dialog box. Assign a model element that represents the partition.

To draw a swimlane on the diagram

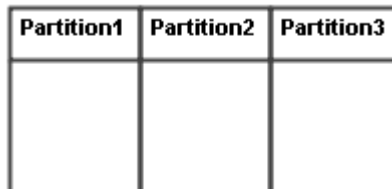
Click the **Swimlane** button on the diagram toolbar and click the diagram.

To set a name for a swimlane

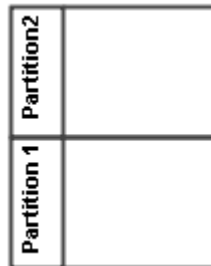
1. Double-click the swimlane line or select **Specification** from the line shortcut menu.
2. The **Activity Partition Specification** dialog box opens. Type a name in the **Name** box.
 - Type the name directly on the Diagram pane.

The Horizontal and Vertical Swimlanes

The following is an example of vertical swimlanes:



The following is an example of horizontal swimlanes:



To add an additional partition

Select a swimlane on the diagram pane and from the shortcut menu, select **Insert Vertical Swimlane** or **Insert Horizontal Swimlane**.

To draw multidimensional swimlanes

1. Draw a vertical swimlane.
2. From the swimlane shortcut menu, select the **Insert Horizontal Swimlane** command.
3. Insert as many horizontal and vertical swimlanes as you need.

The following is an example of the multidimensional swimlanes:

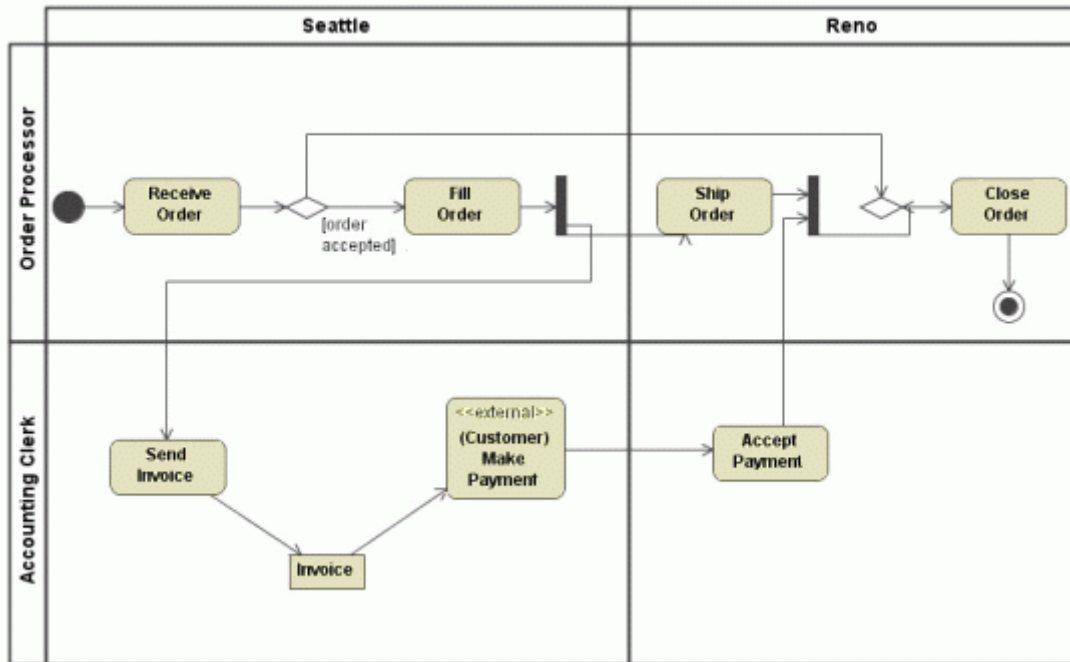
	Seattle	Reno
Order Processor		
Accounting Clerk		

To add model elements to a swimlane

If a swimlane is already drawn in the activity diagram, drawing an action (or any other element) will highlight the swimlane in blue. This means that the action shape will depend on the swimlane symbol.

NOTE: If the model elements depend on a swimlane symbol, they will be deleted if the swimlane symbol is deleted.

The multidimensional swimlanes, with added elements, shown in the Figure below.



Template / Parameterized class

Terms	<p>The OMG UML specification (UML 2.2: Superstructure) states:</p> <p>“A parameterable element is an element that can be exposed as a formal template parameter for a template, or specified as an actual parameter in a binding of a template.”</p> <p>The OMG UML specification (UML 2.2: Superstructure) states:</p> <p>“A templateable element is an element that can optionally be defined as a template and bound to other templates.”</p> <p>The OMG UML specification (UML 2.2: Superstructure) states:</p> <p>“A classifier template parameter exposes a classifier as a formal template parameter.”</p>
--------------	---

To define a parameterized class

1. Double-click the class shape or select **Specification** from the class shape shortcut menu. The **Class Specification** dialog box opens. Click the **Template Parameters** tab.
2. To add or edit a parameterized class, click the **Create** button. The **Select Template Parameter Type** dialog box opens. Select the type from the model tree and click **OK**.
3. To remove the selected parameterized class, click the **Delete** button.

To open the **Classifier Template Parameter Specification** dialog box

In the element **Specification** dialog box tree, expand the **Template Parameters** tab and double click the selected template.

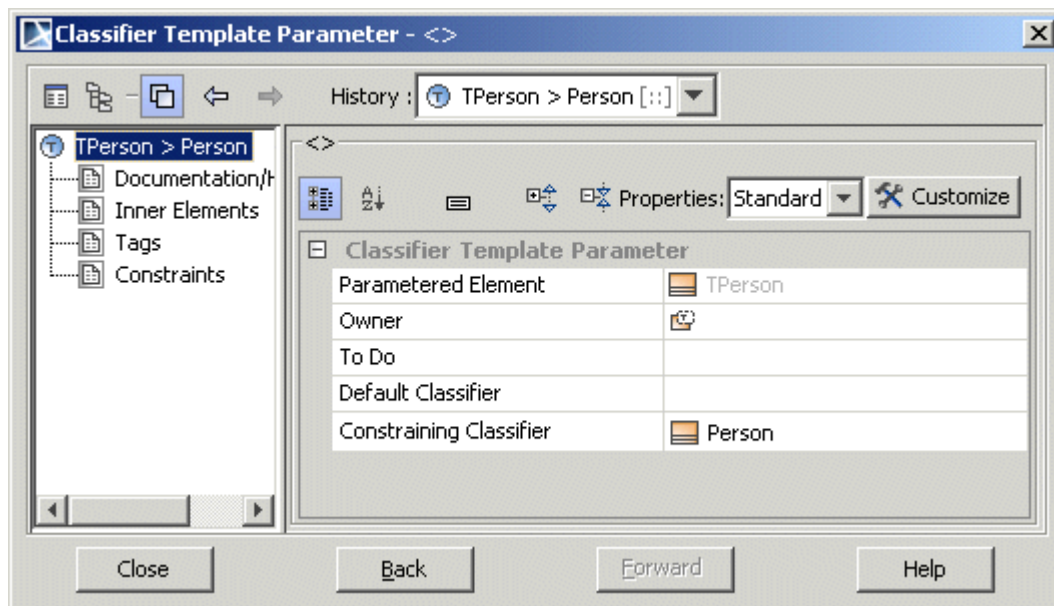


Figure 436 -- Classifier Template Parameter Specification dialog box

Select the **Expert** mode from the **Properties** field to show more properties of the template parameter.

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section

Tab name	Box name	Function
General	Parametered Element	The name of the element, which was parametered by a template parameter.
	Owner	The name of the element, which contains the template parameter.

Transition

A transition is a directed relationship between a source vertex and a target vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to an occurrence of an event of a particular type.

Define the selected transition in the **Transition Specification** dialog box.

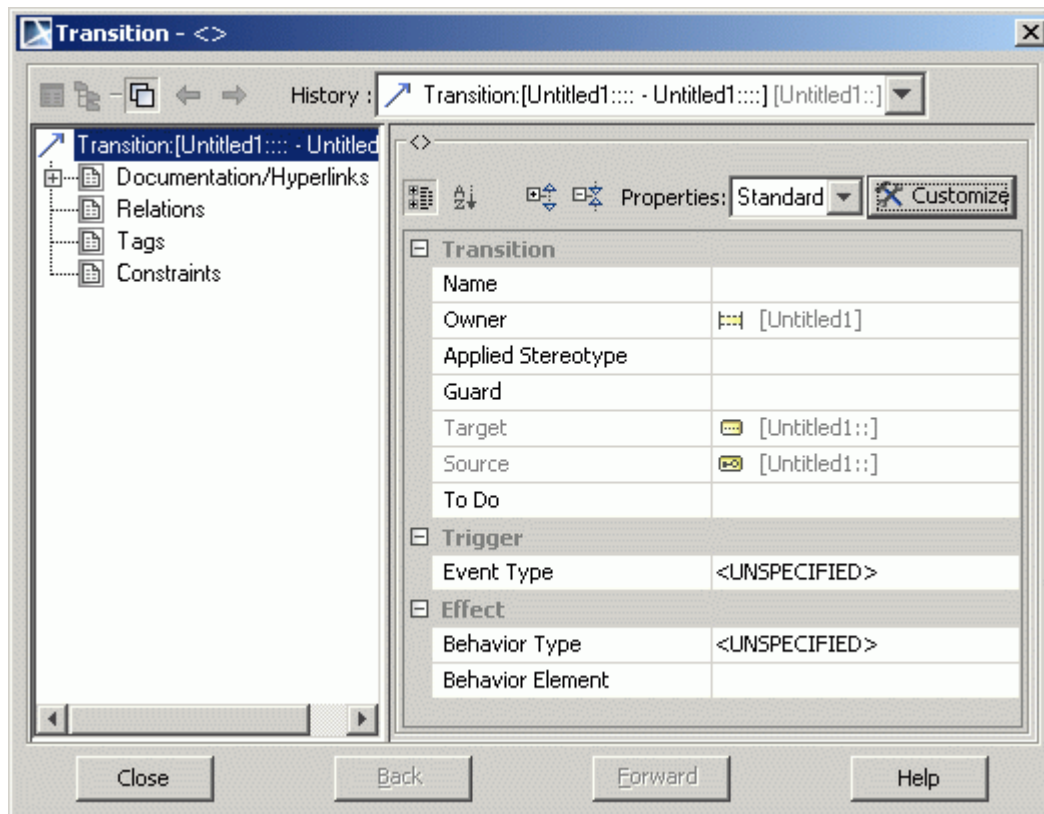


Figure 437 -- Transition Specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about the specification elements not covered in this section.

Tab name	Box name	Function
General Set a general information about the transition.	Guard	A guard is an expression that is attached to a transition as a fine-grained control over its execution. The guard is evaluated when an event instance is dispatched by the state machine.
	Target	The name of the model element, which is the target of the transition.
	Source	The name of the model element, which is the source of the transition.
	Trigger group box	
	Event Type	The type of the event
	Effect group box	
	Behavior Type	The type of the affected action.
	Behavior Element	Click '...' and select an element from the activity, interaction or state machine items list. The corresponding dialog box opens.

To define an event type

- In the **Transition Specification** dialog box, **General** tab, click the arrow down in **Trigger** group box, **Event Type** field.

Event types

Name	Description
Any Receive Event	A transition trigger associated with AnyReceiveEvent specifies that the transition is to be triggered by the receipt of any message that is not explicitly referenced in another transition from the same vertex.
Call Event	A call event represents the reception of a request to cause a specific operation. It is distinct from the call action that caused it. It may cause the invocation of a behavior that is the method of the operation referenced by the call request, if that operation is owned or inherited by the classifier that specified the receiver object.
Change Event	A change event models an event that occurs when an explicit boolean expression becomes true as a result of a change in value of one or more attributes or associations. A change event is raised implicitly and is not the result of some explicit change event actions. An expression specifies the change event: a keyword when followed by an expression.
Creation Event	A creation event models the creation of an object.
Destruction Event	A destruction event models the destruction of an object.
Execution Event	An execution event models the beginning or the end of an execution occurrence.
Send Operation Event	A send operation event specifies the sending of a request to invoke a specific operation on an object.
Send Signal Event	A send signal event specifies the sending of a message to a receiver object.
Signal Event	A signal event represents the reception of a particular (asynchronous) signal. The Signal Event is a child of the Event.
Time Event	A Time Event models the expiration of a specific deadline. Note that the time of occurrence of a time event instance (i.e., the expiration of the deadline) is the same as the time of its reception. An expression specifies the corresponding time deadline: a keyword after, followed by an expression.

Enhanced Event assignment to transition

To add an Event value on a transition can be simply done through the diagram pane:

- **Signal Event.** Type any title on a transition. A signal event is added. The words typed will be the signal name. If such typed signal already exists in the project, and this signal is included in the Signal Event, the Signal Event is assigned to the transition.
- **Call Event.** The name of the operation is displayed on a transition. If an existing operation name is typed on the transition (if this operation is assigned to the Call Event), the Call Event in this operation will be assigned to the transition.
- **Change Event.** To add the Change Event to a transition, type the following: *when ()*. You may add change event expression in the brackets.
- **Time Event.** To add the Time Event to a transition, type the following: *at ()*. You can add the value of the time event in the brackets.

Use Case

A use case represents a typical interaction between a user and a system. It captures some of the functionalities and data that the user works with. It is a type of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies a type of behavior, including the variants, that the subject can perform in collaboration with one or more actors. The subject of a use case could be a physical system or any other element that may have the behavior, such as a component, a sub-system, or a class.

An extension point is a reference to one location within a use case where an action sequences from other use cases can be inserted. Each extension point has a unique name within the use case and a description of the location within the behavior of the use case.

A use case is shown as an ellipse with the the name inside it. The extension points are listed in the Extension Points compartment of the use case.

Define the selected use case in the **Use Case Specification** dialog box.

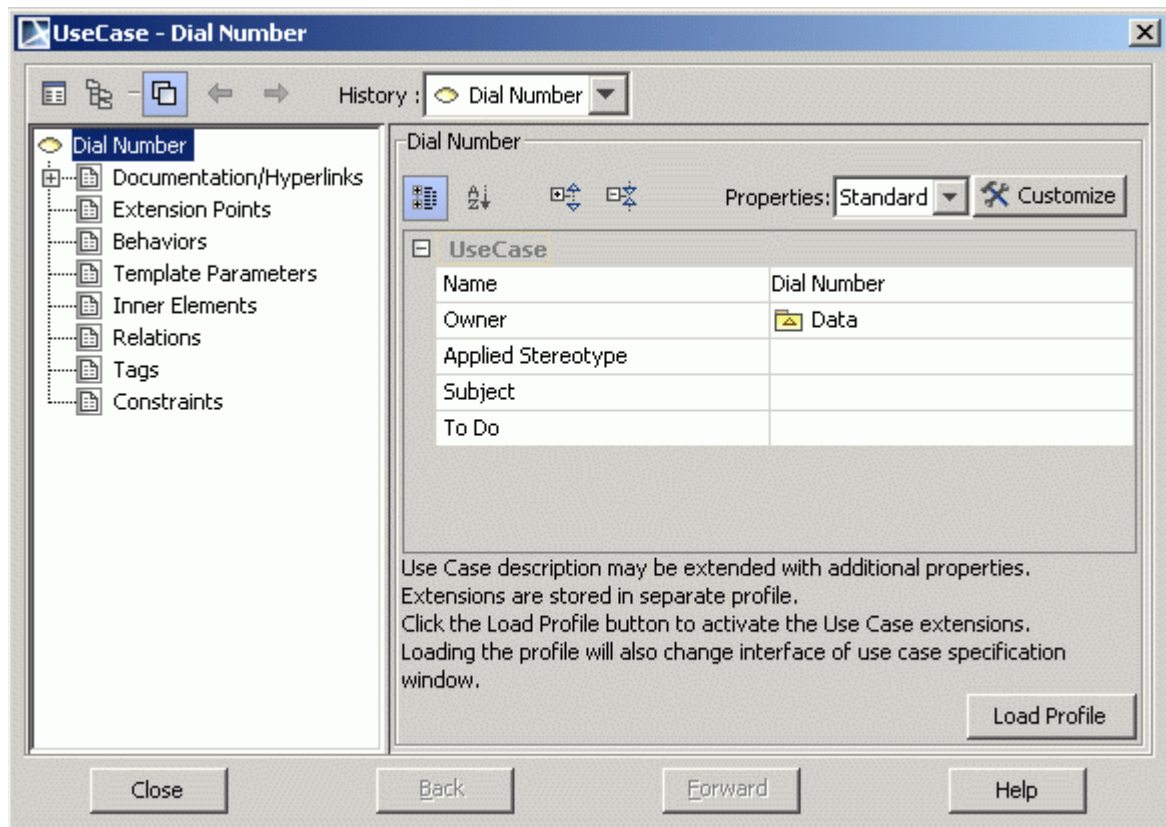


Figure 438 -- Use Case specification dialog box

Refer to the “Specification dialog boxes” on page 325 for information about specification elements..

Group name	Box	Function
General	Load Profile	Loads the Use Case description profile that contains specific extensions.
		NOTE This functionality is available in Standard, Professional, Architect, and Enterprise editions only.

Group name	Box	Function
Behaviors From the drop down list, select an activity, state machine, or interaction to create within a use case.	Type	The type of the created element.
	Name	The name of the created element.
	Create	The list of elements to create opens. Select the element and define the properties in the open properties list.
	Delete	Remove the selected diagram from the use case.
Extension Points	Create	Define a new extension point.
	Delete	Remove the selected extension point from the use case.
Inner elements From the drop down list, choose diagrams or constraints to create as an inner element of the use case.	Name	Type the name for a newly created diagram.
	Type	The type of the created diagram.
	Create	The list of diagrams opens. Select a diagram or constraint and define properties in the open properties list.
	Delete	Remove the selected inner element from the use case.

For more information about working with symbols, see Chapter 5, "Working With Diagrams."

To add an extension point for the use case

1. Open the **Use Case Specification** dialog box.
2. Click the **Extension Points** group.
3. Click **Create**. The **Extension Point** properties window opens. Here you may specify the extension point.
 - Press CTRL+ALT+E.
 - From the use case shortcut menu, select **Insert New Extension Point**.
 - From the use case shortcut menu in the Browser, select **New Element**, and then **Extension Point**. Type the name of the extension point.

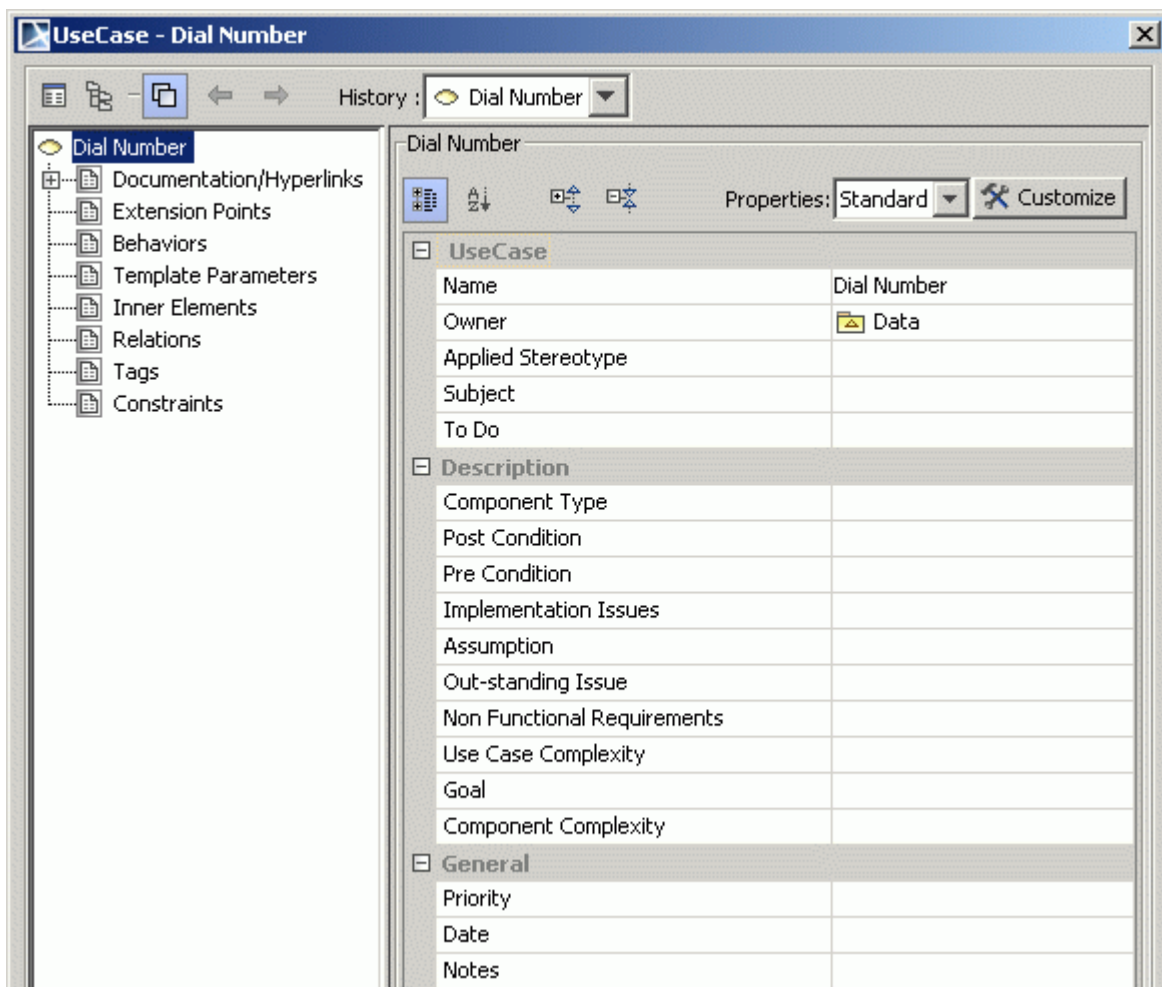
TIPS:

You may create an extension point directly on a diagram: Draw an extension path between two use cases. Then click **YES** in the message window. Once it is created, specify its name.

You may define a use case as abstract (for the detailed description, see "Generalizable elements" on page 900.)

Use Case Extension

MagicDraw provides the UseCase description extensions. Now you may define the pre-conditions and post-conditions for the use cases and other extensions, which are implemented as properties and tags in the **Use Case Specification** dialog box. MagicDraw contains the UseCase Description profile, which can be loaded from the **Use Case Specification** dialog box. The properties are available in the **Use Case** Specification window:



The properties are mapped to tags, which can be listed in the **Specification** dialog box, **Tags** group. If the tags cannot be seen, apply `<<requirementUseCase>>` stereotype to the use case.

IMPORTANT

The tag definitions from the UseCase Description profile are only visible in generated reports, not on the diagrams. For a detailed description about reports, see in “Controlling Merge memory usage” on page 465. For a detailed description of working with tag definitions, see in “Specification dialog boxes” on page 325.

To display the tag definitions from the UseCase Description profile on the diagram

1. Open the UseCase Description profile `UseCase_Profile.xml` as a project (this profile is located in the `<MagicDraw installation directory>/profiles` folder).
2. In the Browser tree select the property, which represents the tag definition and open its **Specification** dialog box.
3. In the **Applied Stereotype** field, remove `<<InvisibleStereotype>>`.
4. Save the project.
5. Reload the profile in the project you are working: from the UseCase Description Profile shortcut menu in the Browser tree, select **Modules**, and then **Reload Module**.

Value Specification

To create a Value Specification using the Browser shortcut menu

Now you can create a standalone Value Specification in a model using the Browser shortcut menu. To create Value Specification from the element shortcut menu in Browser choose **New Element**, **Value Specification** (see Figure 439 on page 1025).

NOTE

Since MagicDraw 16.0 version Value Specifications are displayed in the Containment tree of the Browser.

11 MODEL ELEMENTS

Value Specification

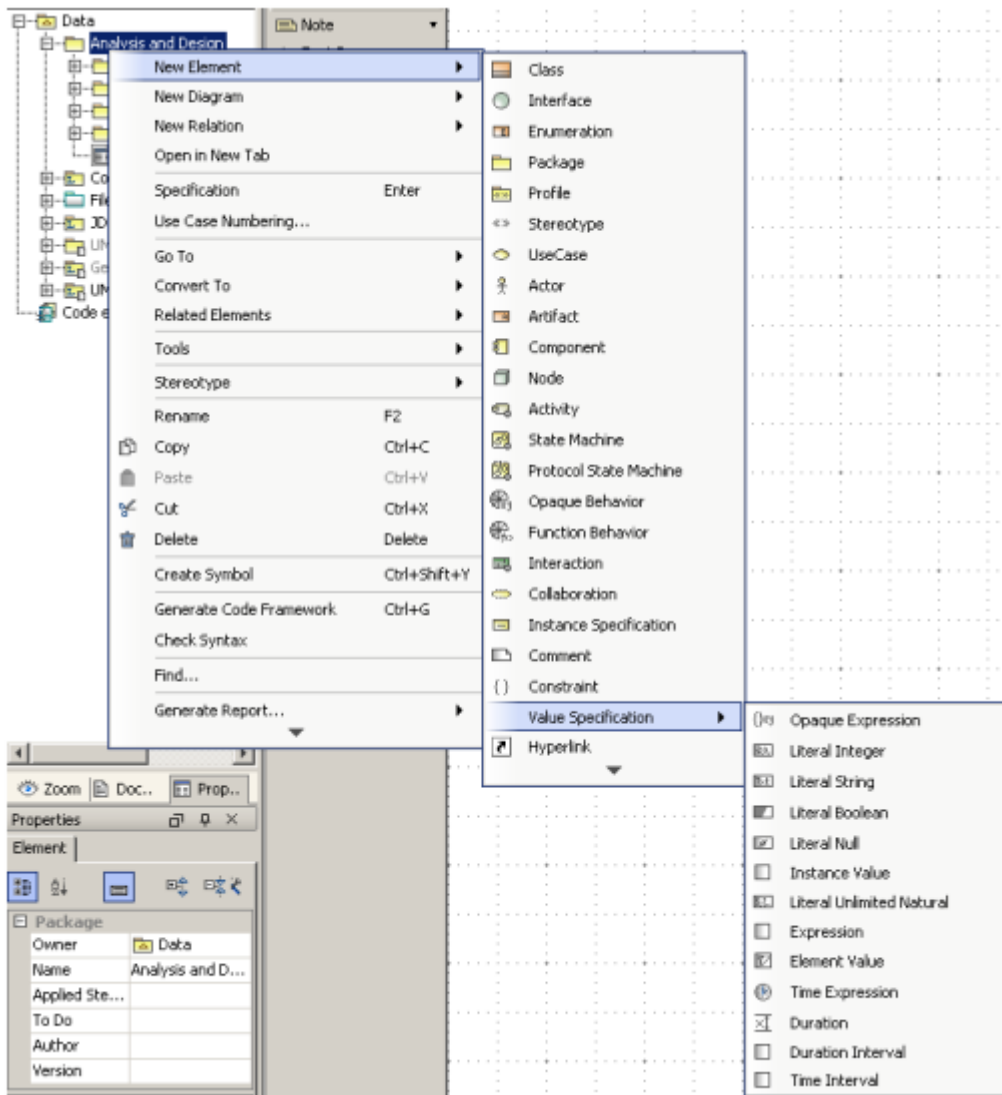


Figure 439 -- Creating Value Specification from Browser

APPENDIX: LIST OF ICONS

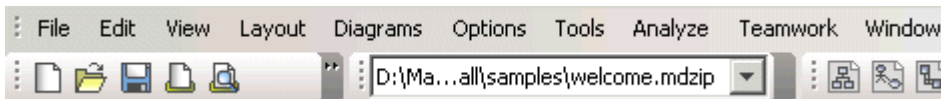
This document includes icons and buttons used in the MagicDraw GUI (main toolbars and Browser Tree) and Icons of modules and Profiling mechanism.

- “Icons from the MagicDraw GUI” on page 1026.
- “Diagram Icons” on page 1039.
- “Icons of general elements” on page 1041.
- “Icons of relationships” on page 1055.
- “Icons from Modules and Profile mechanism” on page 1061.

Icons from the MagicDraw GUI

Main Toolbar







Speed up your work using the MagicDraw main toolbar. It is located at the top of the MagicDraw window, below the main menu.



For more information about working with toolbars, see “Toolbars” on page 102.



APPENDIX: LIST OF ICONS

Icons from the MagicDraw GUI

Toolbar button	Title	Description
File group		
	New Project (Ctrl+N)	To create a new blank project, press the New Project button: <ul style="list-style-type: none">• The New Project dialog box opens. Select the Blank Project icon.• Specify the file name in the Name text box.• Click the “...” button to select a location to store the newly created project in your computer. Click OK.
	Open Project (Ctrl+O)	To open an existing project, press the Open Project button.
	Save Project (Ctrl+S)	To save the current project.
	Print Active Diagram (Ctrl+P)	To print an open diagram.
	Print Preview	The Print Preview dialog box opens showing how your diagram looks before printing.
	Find (Ctrl+F) Find TODO Quick Find (Ctrl+Alt+F)	Search for an element, symbol or diagram in the project according to your selected criteria.


APPENDIX: LIST OF ICONS

Icons from the MagicDraw GUI

Toolbar button	Title	Description
	Undo action (Ctrl+Z)	<p>Undo the last action you performed while drawing the diagram on the Diagram pane (moving, dragging, resizing, cutting, copying, pasting, deleting, selecting, editing shapes, setting project and shape properties, etc.). Actions are reversed in the order you have performed them.</p> <p>The Undo command is unavailable until you perform an action after loading an existing project or creating a new project. By default, the limit of the undo mechanism is 100 steps backwards.</p> <p>To change the limit, select Environment from the Options menu. The Environment Options dialog box opens. Change the Undo List Size property.</p> <p>Each command has an easily recognized name. You will be able to see the command history and undo or redo action history. The main window will have two lists of commands: one for the undo commands, another one for the redo commands.</p>
	Redo action (Ctrl+Y)	<p>Restore the Undo action (moving, dragging, resizing, cutting, copying, pasting, deleting, selecting, etc.).</p> <p>The Redo command is unavailable until you use the Undo command.</p>
Open Projects group		A combo box displaying a list of open projects.


APPENDIX: LIST OF ICONS



Icons from the MagicDraw GUI

Toolbar button	Title	Description
	List of open projects	You will see the name and location of the currently open project. To see a list of all open projects, click the small arrow on the right side of the drop-down list box.

Buttons from the diagram list toolbar

To work with diagram windows and tooltips of model elements, use the buttons on the main toolbar:

Previous Diagram, Next Diagram or Tooltips Style 

Toolbar button	Title	Description
	Previous Diagram (Alt+Left)	Opens the previously open diagram.
	Next Diagram (Alt+Right)	Opens the next diagram.

APPENDIX: LIST OF ICONS

Icons from the MagicDraw GUI


Toolbar button	Title	Description
	ToolTips Style button	<p>Click the ToolTips Style button and select the kind of tool tips you wish to display in symbols on the diagram pane:</p> <ul style="list-style-type: none">• Do not show – do not show tips (default).• Object name – show the symbol name and location path from the Data package in the following style: <Element Name> [<path>]• Object Documentation – show the documentation associated with the desired symbol. <p>NOTE To see the tool tips, drag the cursor over the desired symbol on the diagram.</p>

Diagram main toolbar

The diagram toolbar contains buttons for working with symbols on the diagram pane. Select any symbol or path on the diagram pane and the required buttons from the diagram main toolbar become active.











Use the diagram main toolbar to change the symbol layout, path style, symbol properties style, diagram zoom as well as symbol copy/paste, cut, or delete actions.

Toolbar button	Title	Description
<u>Layout</u> group		

APPENDIX: LIST OF ICONS









Icons from the MagicDraw GUI

	Quick diagram layout	<p>Apply the recommended layout tool with default options to the active diagram.</p> <p>Press the small arrow near the Quick Diagram Layout button to see other available layouts.</p>
	Make same width	Layout the selected shapes according to their width. After the layout, the same width will be apply to the shapes (according to the widest).
	Make same height	Layout the selected shapes according the their height. After the layout, the same height will be applied the the shapes (according to the highest).
	Make same size	Layout the selected shapes according to their width and height. After the layout, the same width and height will be applied to the shapes according to the widest and highest shape.
	Center Horizontally	Center the selected shapes on a horizontal line.
	Center Vertically	Center the selected shapes on a vertical line.
	Space Evenly Horizontally	Space the selected shapes evenly. The space between the selected shapes is equally distributed on a horizontal line.
	Space Evenly Vertically	Space the selected shapes evenly. The space between the selected shapes is equally distributed on a vertical line.

Paths Editing group


APPENDIX: LIST OF ICONS

Icons from the MagicDraw GUI

	Remove Break Points	To remove all angles of a path, press the Remove Break Points button.
	Rectilinear	To change a path style to rectilinear lines, press the Rectilinear button.
	Reset Labels Positions	To reset the path labels to the default position, press the Reset Labels Position button.
<u>Symbol Editing group</u>		
	Fill Color	Press the button to apply the selected fill color or press the small arrow near the button to select another color.
	Set Selected Symbol Style as Default	If a new style was set, it will be applied for all newly created elements after drawing them on the diagram pane.
	Apply Default Symbol Style	The selected symbol style will be changed to it's default style. (The default symbol style is defined in the Project Options dialog box, Symbol property styles branch).
	Select All of the Same Type	All shapes of the same style are selected, for example, all classes will be selected.
<u>View group</u>		
	Zoom In, Zoom Out, and Percentage drop down box	Using the view group buttons, you can change the view of the diagram by zooming it in and out. NOTE You can also change the diagram view in the Browser, Zoom panel.
<u>Edit group</u>		

APPENDIX: LIST OF ICONS

Icons from the MagicDraw GUI

	Cut, Copy, Paste and Delete	Using the Edit group buttons, you can copy or delete symbols from the diagram pane. NOTE After a symbol is deleted from the diagram pane, it does not mean that the element is deleted from the project.
---	-----------------------------	--

Buttons from the Browser window

The Browser is a hierarchical navigational tool that allows you to manage your model data, including packages, components, classes, all UML diagrams, extension mechanisms, and other data.

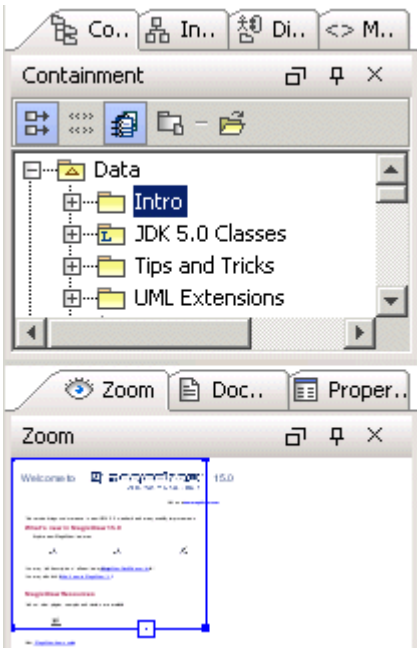
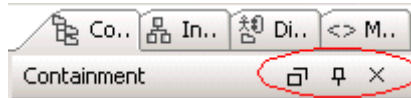







Figure 1 -- Browser window

Floating, Auto-hide and Close buttons

To each Browser window there are added Toggle Floating, Toggle auto-hide, and Close buttons. You can move or hide a window using these buttons . See the circled buttons in the image below.



Icon	Title	Description
	Toggle Floating	Press the Toggle Floating button, the window is split and you can move the window to any desired position.
	Toggle auto-hide	Press the Toggle auto-hide button, the current Browser window is hidden. The tab of the hidden window is displayed on the left side of MagicDraw (in vertical position). Bring the mouse over the hidden window tab and the window is displayed.
	Close	Close the current window.
	Toggle Floating	The floating window is docked again.
	Toggle auto-hide	The auto-hide enabled window is displayed again.


The browser consists of two parts:

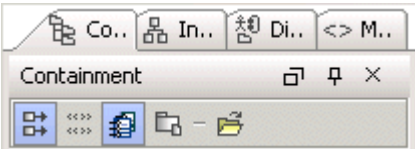
1. Containment tree/Inheritance tree/Diagrams tree/Model Extensions tree/Search Results part.
2. Zoom/Documentation/Properties window.






APPENDIX: LIST OF ICONS

Icons from the MagicDraw GUI


Buttons from the Containment tree

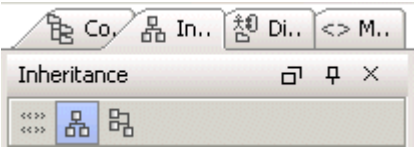
In the Browser, the Containment tree  is active by default. To open the Containment tree, click the Containment Tree tab at the top of the Browser. The Containment tree displays a model data, and groups it in logical sets.





Icon	Title	Description
	Show Full Types in Browser	To show/hide full information of the operations, attributes, and relationships in the Containment Tree.
	Show Stereotypes	To show/hide stereotypes near the element name.
	Show Code Engineering Sets	To show/hide Code Engineering Sets branch in the Browser.
	Show Auxiliary Resources	To show/hide modules and profiles in the Browser.
	Open in New Tab	A new tab with a package name and tree content will be opened in the Browser.

Buttons from the Inheritance tree

To open the Inheritance tree  , click the Inheritance Tree tab at the top of the Browser. The Inheritance tree represents the classifiers, packages, data types, stereotypes hierarchy of your project. The inheritance according to the UML Specification is shown using a generalization relationship.




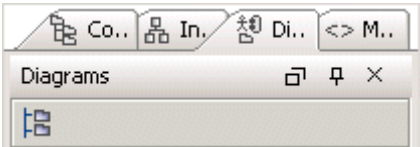
Icon	Title	Description
	Show only hierarchies	The generalization hierarchies are displayed in the tree. If a classifier has no generalization relationship, it will not be visible on the tree.
	Invert Tree	The hierarchy of general classifiers and children is displayed in the Inheritance tree. After inverting the tree, the classifier tree view will be changed, making the child a root classifier.


APPENDIX: LIST OF ICONS

Icons from the MagicDraw GUI


Buttons from the Diagrams tree

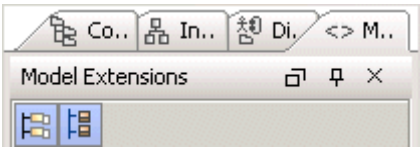
To open the Diagram tree  , click the Diagram Tree tab at the top of the Browser. The Diagram tree in the Browser represents the external structure of a diagram.





Icon	Title	Description
	Group by Diagram Type	If the Group by Diagram Type button is pressed, diagrams are listed in the packages by diagram type.

Buttons from the Model Extensions tree

To open the Model Extensions tree  , click the Model Extensions tree tab at the top of the Browser. The Model Extensions Tree contains all Stereotypes that are predefined and created manually in the project.




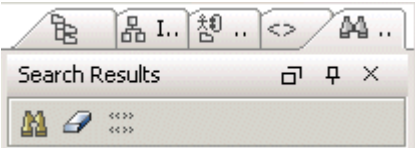
Icon	Title	Description
	Group by Profiles	If the Group by Profiles button is pressed, groups of profiles and the elements inside them are listed.
	Group by Metaclass	If the group by Metaclass button is pressed, groups of Metaclasses are displayed as packages and the elements are listed inside them.




APPENDIX: LIST OF ICONS

Icons from the MagicDraw GUI

Buttons from the Search Results tree

To open the Search Results tree, click the  from the **Edit** main menu, select the **Find** command. The **Search Results** tree shows results of the search.











Icon	Title	Description
	Find	Press the Find button and the Find dialog box opens.
	Clear Results	Press the Clear Results button to remove elements from the Search Results tree.
	Show Stereotypes	Press the Show Stereotypes button to display stereotypes beside the element.

APPENDIX: LIST OF ICONS










Diagram Icons

Diagram Icons

Icon	Diagram	Description
	Class diagram	A class diagram is a graphic representation of the static structural model. It shows classes and interfaces, along with their internal structure and relationships. The classes represent the types of objects that are handled in a system. ^a
	Use Case diagram	A use case is a description of the functionality (a specific usage of a system) that a system provides.
	Collaboration diagram	A collaboration diagram illustrates the various static connections between objects and it models their interactions.
	Sequence diagram	A sequence diagram is a time-oriented view of the interaction between objects.
	State diagram	A state diagram describes the lifecycle of an object and its behavior (such as a procedure or operation), or a behavior feature (such as a use case).
	Protocol State Machine diagram	A protocol state machine is always defined in the context of a classifier. It specifies which operations of the classifier can be called in which state and under which condition, thus specifying the allowed call sequences on the classifier operations.*
	Activity diagram	The purpose of an activity diagram is to focus on flows driven by the internal processing (as opposed to external events).
	Implementation diagram	The Implementation diagrams help developers describe the logical software structure inside a computer system or across a large distributed system.
	Composite Structure diagram	A composite structure diagram allows a decomposition and modeling of the internal structure of classifiers.







APPENDIX: LIST OF ICONS

Diagram Icons

Icon	Diagram	Description
	Business Process diagram	The Business Process Diagrams convey business processes of different participants in a modeled system. In a MagicDraw model, the Process contains a Business Process Diagram, depicting it as a graph of Flow Objects.
	CORBA IDL diagram	CORBA IDL diagram facilitates the creation of CORBA IDL elements. Also the following patterns are available for CORBA IDL: Interface, Value Type, Type Definition, Sequence, Array, Fixed, Union, Enumeration, Struct, and Exception.
	Free Form diagram	In the free form diagram you may draw all types of element shapes and also describe the business workflow.
	Generic DDL diagram	A generic DDL diagram is used to draw database definition elements. The Generic DDL diagrams simplify the creation of primary key, foreign key, triggers, and so forth.
	Networking diagram	A networking diagram allows a visual display of a network topology. The Networking Profile contains stereotypes for a network description.
	Oracle DDL diagram	Use the Oracle DDL diagram to draw database definition elements. This diagram is based on the Generic DDL diagram.
	Robustness diagram	The robustness diagram represents robustness analysis. It includes elements from the class diagram and actors from the use case diagram with predefined stereotypes.
	Struts diagram	A struts diagram is an extension of the UML notation. It contains the same model elements which belong to the class diagram, except they have predefined stereotypes.
	Time diagram	A time diagram is similar to a sequence diagram, except the model elements of the time diagram have predefined stereotypes.

APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Diagram	Description
	WSDL diagram	A WSDL diagram is used to draw WSDL elements. It enables you to create all the elements used in a wsdl file.
	Web diagram	A web system consists of server applications, network, communicating protocol, and the browser.
	XML Schema diagram	The purpose of this diagram is to create the structure of an xml schema file. The Diagram allows a quick drawing of xml schema elements.
	oAW Metamodeling diagram	<p>The openArchitectureWare (oAW) is a popular model driven development & architecture framework based on Eclipse. This framework consists of various components, which allow you to manipulate models in various ways.</p> <p>The oAW Metamodeling diagram is used together with the oAW's uml2ecore transformer to simplify the meta-model preparation.</p>
	Content Diagram	The purpose of the content diagram is to generate or represent a project structure (diagrams). The relations between diagrams are represented. The content table works as a table of contents of a project.
	Dependency Matrix	The Dependency Matrix is a method of visualizing and representing dependency criteria. Diagrams, UML and extended UML elements serve as row and column entries. The cells in the matrix show where these elements are associated or related.

a) Material from the OMG UML Specification has been duplicated with permission.









Icons of general elements

This section lists all element icons displayed in the MagicDraw Browser window. To make your icons search easier, the icons of relationships are listed in the next section “Icons of

APPENDIX: LIST OF ICONS










Icons of general elements

relationships” on page 1055, and the icons used in modules/profiling mechanism are listed in the section “Icons from Modules and Profile mechanism” on page 1061.

Icon	Title	Description
	Abstract class	An abstract class is a class, which represents the conceptual set of the classifiers. In the Class specification dialog box select the Is Abstract option and a class icon changes to the icon of the abstract class.
	Accept Event Action	An accept event action is an action that waits for the occurrence of an event that meets the specified conditions. ^a
	Action	An action is a named element that is the fundamental unit of an executable functionality. The execution of an action represents some transformation or processing in the modeled system, be it a computer system or otherwise.*
	Activity	An activity element is created on the activity diagram creation.
	Activity Parameter Node	An activity final node is a final node that stops all flows in an activity.*
	Actor	An actor represents the roles played by the human users, external hardware, and other subjects.*
	Artifact	An artifact represents a physical piece of information that is used or produced by a software development process.*
	Artifact Instance	An instance of an artifact.










APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Association Class	An Association Class can be seen as an association that also has class properties, or as a class that also has association properties. Not only it connects a set of classifiers, but also defines a set of features that belong to the relationship itself, not to any of the classifiers.*
	Attribute	An attribute is a named property of a class that describes a range of values that can be held by instances of that class.
	Call Operation Action	The call operation action transmits an operation call request to the target object, where it may cause the invocation of an associated behavior.*
	Central Buffer Node (Object Node)	An object node is an activity node that indicates an instance of a particular classifier, possibly in a particular state, may be available at a particular point in the activity.*
	Choice	The choice vertices, when reached, result in the dynamic evaluation of the guards of the triggers of its outgoing transitions.*
	Class	A class is the descriptor for a set of objects with similar structure, behavior, and relationships.
	Collaboration	A collaboration is represented as a kind of classifier and it defines a set of cooperating entities to be played by instances (its roles) as well as a set of connectors that define communication paths between the participating instances.*
	Collaboration Use	A collaboration use represents a particular use of collaboration to explain the relationships between the properties of a classifier.*
	Combined Fragment	A combined fragment defines an expression of interaction fragments. It is defined by an interaction operator and the corresponding interaction operands.*







APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Comment	A comment gives an ability to display different remarks on diagrams. * It may be attached to multiple elements.
	Component	A component represents all kinds of elements that pertain to piecing together software applications. They can be simple files, such as DLLs or executables.
	Component Instance	An instance of a component that may reside on a node instance.
	Composite State	A composite state is a state with one region. There is a set of two or more states placed within a region. These are known as substates. The substates describe states within a state.
	Orthogonal State	An orthogonal state is a state with two or more regions. In each region, you may draw one life cycle with both a beginning and an end.
	Conditional Node	A conditional node is a structured activity node that represents an exclusive choice among some number of alternatives. *
	Connection Point Reference	A connection point reference represents the use of entry/exit points.
	Constraint	A constraint is a condition or restriction expressed in a natural language text or in a machine readable language for the purpose of declaring some of the semantics of an element. *
	Data Store	A data store keeps all tokens that enter it, copies them when they are chosen to move downstream. Incoming tokens containing a particular object replace any tokens in the object node containing that object. *



APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Data Type	A data type is a type whose instances are identified only by their value.*
	Deep History	A deep history represents the most recent active configuration of the composite state that directly contains this pseudostate.*
	Decision/Merge Node in activity diagram Pseudo State in a state diagram	Decisions are made using guard conditions. They help protect transitions that depend on a guarding condition. The pseudo states are typically used to connect multiple transitions into more complex state transitions paths.*
	Deployment	A node deploys and provides a place to store and/or execute an artifact.
	Deployment Specification Instance	An instance of a deployment specification element.
	Device	A device is a physical computational resource. For example; a piece of hardware such as a desktop computer, a processor, a server, or a human work unit such as a department or team.
	Device Instance	An device instance is an instance of a device.
	Duration	A duration defines a value specification that specifies the temporal distance between two time instants.*
	Duration Constraint	A duration constraint defines a constraint that refers to a duration interval.*
	Element Value	A value specification is an abstract metaclass used to identify a value or values in a model. It may make reference to an instance or it may be an expression denoting an instance or instances when evaluated.*









APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Entry Point	An entry point connection point reference as the target of a transition implies that the target of the transition is the entry point pseudostate as defined in the submachine of the submachine state.*
	Enumeration	An enumeration is a kind of data type, whose instances may be any of a number of user-defined enumeration literals.
	Enumeration Literal	An enumeration literal defines an extension element of an enumeration data type.
	Event (all type of events)	An event is the specification of some occurrences that may potentially trigger effects by an object.*
	Execution Environment	An execution environment is a node that offers an execution environment for specific types of components that are deployed on it in the form of executable artifacts.*
	Execution Environment Instance	An instance of execution environment.
	Exit Point	An exit point connection point reference as the source of a transition implies that the source of the transition is the exit point pseudostate as defined in the submachine of the submachine state that has the exit point connection point defined.*
	Expansion Node	An expansion node is an object node used to indicate a flow across the boundary of an expansion region.*
	Expansion Region	An expansion region is a structured activity region that executes multiple times corresponding to the elements of an input collection.*











APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Final State	A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine is completed.*
	Activity Final	An activity may have more than one activity final node. The first one reached stops all flows in the activity.*
	Function Behavior	A function behavior is an opaque behavior that does not access or modify any objects or other external data.*
	Hyperlink	An icon with a small arrow image on the left-bottom side indicates that element has a hyperlink to another element/symbol, file, or web page. For more information about hyperlinks, see “Defining Hyperlinks Between Elements” on page 362.
	Information Flows	The InformationFlows package provides mechanisms for specifying the exchange of information between entities of a system at a high level of abstraction.*
	Information Item	An information item is an abstraction of all kinds of information that can be exchanged between objects.*
	Initial Node	An initial node is a control node at which a flow starts when the activity is invoked.*
	Input Pin	An input pin is a pin that holds input values to be consumed by an action.*
	Output Pin	An output pin is a pin that holds output values produced by an action.*
	Instance	An instance specification is a model element that represents an instance in a modeled system.*

APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Interaction	An interaction is a unit of behavior that focuses on the observable exchange of information between connectable elements.*
	Interaction Operand	An interaction operand is contained in a combined fragment. It represents one operand of the expression given by the enclosing combined fragment.*
	Interaction Use	An interaction use refers to an Interaction. The InteractionUse is a shorthand for copying the contents of the referred Interaction where the interaction use is.*
	Interface	An interface is a specifier for the externally-visible operations of a class, component, or other classifier (including subsystems) without a specification of the internal structure.
	Interruptible Region	An interruptible region contains activity nodes. When a token leaves an interruptible region via edges designated by the region as the interrupting edges, all tokens and behaviors in the region are terminated.*
	Junction	The junction pseudo state corresponds to the merge and the static conditional branch.
	Lifeline	A lifeline represents an object. The lifeline runs from the beginning of the interaction, at the top of the diagram, to the end of the interaction at the bottom of the line.
	Literal Boolean	A literal boolean is a specification of a boolean value.*
	Literal Integer	A literal integer is a specification of an integer value.*
	Literal Null	A literal null specifies the lack of value.*









APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Literal String	A literal string is a specification of a string value.*
	Literal Unlimited Natural	A literal unlimited natural is a specification of an unlimited natural number.*
	Loop Node	A loop node is a structured activity node that represents a loop with the setup, test, and body sections.*
	MetaClass	A class whose instances are classes. Metaclasses are typically used to construct metamodels.
	Method (Operation)	An operation is a behavioral feature of a classifier that specifies the name, type, parameters, and constraints for invoking an associated behavior.*
	Model	A model is an abstraction of a physical system from a particular point of view.
	System Boundary	A system boundary element consists of use cases related by the exclude or include (uses) relationships.
	Model Library	<p>This icon depicts Model with applied Model Library stereotype.</p> <p>The model library is a package that contains model elements that are intended to be reused by other packages.*</p>
	N-ary Association	An n-ary association is an association among two or more classes (a single class may appear more than once).
	Node	<p>A node is a computational resource upon which artifacts may be deployed for execution.</p> <p>The nodes can be interconnected through communication paths to define the network structures.*</p>










APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Node Instance	A node instance is an instance of a node where the component instances may reside.
	Opaque Action	An opaque actions is an action with implementation-specific semantics.*
	Opaque Behavior	A behavior with implementation-specific semantics.* The Opaque Behavior is introduced for implementation-specific behavior or for use as a place-holder before one of the other behaviors is chosen.*
	Opaque Expression	An opaque expression is an uninterpreted textual statement that denotes a (possibly empty) set of values when evaluated in a context.*
	Package Model Library	This icon depicts a Package with applied Model Library stereotype. The model library is a package that contains model elements that are intended to be reused by other packages.*
	Package	A package groups classes and other model elements together.
	Parameter	A parameter is a specification of an argument used to pass the information on to or out of an invocation of a behavioral feature.*
	Port	A port is a property of a classifier that specifies a distinct interaction point between that classifier and its environment or between the (behavior of the) classifier and its internal parts.*
	Primitive Type	A primitive type defines a predefined data type, without any relevant substructure.*







APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Protocol State Machine	A protocol state machine is always defined in the context of a classifier. It specifies which operations of the classifier can be called in which state and under which condition, thus specifying the allowed call sequences on the classifier's operations.*
	Signal Reception (Reception)	A reception is a declaration stating that a classifier is prepared to react to the receipt of a signal.*
	Region	A region is an orthogonal part of either a composite state or a state machine. It contains states and transitions.*
	Send Signal Action	A send signal action is an action that creates a signal instance from its inputs, and transmits it to the target object, where it may cause the discharge of a state machine transition or the execution of an activity.*
	Sequence Node	A sequence node is a structured activity node that executes its actions in order.*
	Shallow History	A shallow history represents the most recent active substate of its containing state.*
	Shared Package	<p>Not all module contents are visible in the using project. A module has a shared part and a private part. Only the contents of the shared part are visible in the working project.</p> <p>The shared packages are marked with a hand image. For more information about project partitioning, see "Project Partitioning" on page 200.</p>
	Signal	A signal is a specification of send request instances communicated between objects.*
	Slot	A slot specifies that an entity modeled by an instance specification has a value or values for a specific structural feature.*









APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	State Machine	A state machines can be used to express the behavior of part of a system.*
	State	A state specifies how the object reacts to events occurring around it.
	Stereotype	A stereotype is an extension mechanism that defines a new and more specialized element of the model based on an existing element.
	Structured Activity Node	A structured activity node is an executable activity node that may have an expansion into the subordinate nodes as an activity group.*
	Submachine State	A submachine state specifies the insertion of the specification of a submachine state machine.*
	Subsystem	A subsystem is a unit of hierarchical decomposition for large systems.*
	Subsystem Instance	An instance of the subsystem.
	Swimlane	Actions and subactivities can be organized into a Swimlane in the activity diagrams. The swimlanes are used to organize responsibility for actions and subactivities according to a class.
	Fork/Joint Horizontal or Vertical	<p>A fork node is a control node that splits a flow into multiple concurrent flows.*</p> <p>A joint node is a control node that synchronizes multiple flows.*</p>
	Template Parameter	A template parameter exposes a parameterable element as a formal template parameter of a template.*


APPENDIX: LIST OF ICONS

Icons of general elements

Icon	Title	Description
	Template Parameter Substitution	A template parameter substitution relates the actual parameter(s) to a formal template parameter as part of a template binding.*
	Template Parameter Signature	A template signature bundles the set of formal template parameters for a templated element.*
	Time Constraint	A time constraint specifies the combination of min and max timing interval values.
	Time Expression	A time expression defines a value specification that represents a time value.*
	Time Observation	An time observation is a reference to a time instant during an execution. It points out the element in the model to observe and whether the observation is made when this model element is entered or when it is exited.*
	Time Event	A time event specifies a point of time by an expression. The expression might be absolute or might be relative to some other point of time.*
	Trigger	<p>A trigger specification may be qualified by the port on which the event occurred.*</p> <p>A trigger relates an event to a behavior that may affect an instance of the classifier.*</p>
	Use Case Instance	A use case instance is an instance of a use case.
	Use Case	A use case is a kind of behavior-related classifier that represents a declaration of an offered behavior.*
	Value Pin	A value pin is an input pin that provides a value by evaluating a value specification.*

APPENDIX: LIST OF ICONS

Icons of general elements









Icon	Title	Description
	Variable	A variable is considered as a connectable element.*

a) Material from the OMG UML Specification has been duplicated with permission.

APPENDIX: LIST OF ICONS










Icons of relationships

Icons of relationships

Icon	Title	Description
	Relations branch	Most of the relationships are included in the Relations branch in the Browser.
	Abstraction	An abstraction is a dependency relationship that relates two elements or sets of elements that represent the same concept at different levels of abstraction or from different viewpoints. ^a
	Aggregation association	An aggregation describes a special type of association designed to help cope with complexity.
	Assembly Connector	An assembly connector is a connector between two components that defines that one component provides the services that another component requires.*
	Association	An association answers to the question why two classes of objects need to know about one another.
	Communication Path	The communication path is a subclass of association. It specifies the relationship between nodes by defining the number of nodes that may be connected (multiplicity), and the nature of the connection, via the name of the path or a stereotype.
	Association Class	An association class can be seen as an association that also has class properties, or as a class that also has association properties. Not only it connects a set of classifiers, but also defines a set of features that belong to the relationship itself, not to any of the classifiers.*
	Call Message	A call message represents the request to invoke a specific operation.
	Composition	A composition is used for aggregations where the life span of the member object depends on the life span of the aggregate.




APPENDIX: LIST OF ICONS

Icons of relationships

Icon	Title	Description
	Connector	A connector specifies a link that enables communication between two or more instances.*
	Constraint	A constraint is a condition or restriction expressed in a natural language text or in a machine readable language for the purpose of declaring some of the semantics of an element.*
	Control Flow	A control flow is an edge that starts an activity node after the previous one is finished.*
	Create Message	A create message is message designating the creation of another lifeline object.*
	Delegation Connector	A delegation connector is a connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component parts.*
	Dependency	A dependency indicates a semantic relationship between two model elements (or two sets of model elements).
	Deployment	A deployment is a relationship between the location and the artifact.
	Delete Message	A delete message is message designated to terminate another lifeline.*
	Direct Association	A directed relationship represents a relationship between a collection of source model elements and a collection of target model elements.*








APPENDIX: LIST OF ICONS

Icons of relationships

Icon	Title	Description
	Directed Aggregation Association	<p>An aggregation describes a special type of association designed to help cope with the complexity.</p> <p>A directed relationship represents a relationship between a collection of source model elements and a collection of target model elements.*</p>
	Direct Composition Association	<p>A composition is used for aggregations where the life span of the member object depends on the life span of the aggregate.</p> <p>A directed relationship represents a relationship between a collection of source model elements and a collection of target model elements.*</p>
	Element Import	<p>An element import is defined as a directed relationship between an importing namespace and a packageable element.*</p>
	Exception Handler	<p>An exception handler is an element that specifies a body to execute in case the specified exception occurs during the execution of the protected node.*</p>
	Extend	<p>A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case.*</p>
	Extension	<p>An extension point identifies a point in the behavior of a use case where that behavior can be extended by the behavior of some other (extending) use case, as specified by an extend relationship.*</p>
	Generalization	<p>A generalization is the relationship from the child element (the more specific element, such as a subclass) to the parent (the more general element, such as a super class) that is fully consistent with the first element and that provides additional information.</p>








APPENDIX: LIST OF ICONS

Icons of relationships

Icon	Title	Description
	Generalization Set	A generalization set defines a particular set of generalization relationships that describe the way in which a general classifier may be divided using specific types. *
	Include	An include (uses) relationship from the use case A to the use case B indicates that an instance of the use case A will also contain the behavior as specified by B. The behavior is included at the location which is defined in A.
	Information Flow	An information flow specifies that one or more information items circulates from its sources to its targets. *
	Interface Realization	An interface realization is a specialized realization relationship between a behavioral classifier and an Interface. *
	Link	An instance specification whose classifier is an association represents a link and is shown using the same notation as for an association, but the solid path or paths connect the instance specifications rather than the classifiers. *
	Manifestation	A manifestation is the concrete physical rendering of one or more model elements by an artifact. *
	Package Merge	A package merge is a directed relationship between two packages that indicates that the contents of the two packages are to be combined. *







APPENDIX: LIST OF ICONS

Icons of relationships

Icon	Title	Description
	Message	A message is a named element that defines one specific kind of communication in an Interaction. A communication can be, for example, raising a signal, invoking an operation, creating or destroying an Instance. The message specifies not only the kind of communication given by the dispatching execution specification, but also the sender and the receiver.*
	Send Message	The message was generated by an asynchronous send action.*
	Non-navigable Association	An association with non-navigable association ends.
	Object Flow	An object flow is a technique used to capture how objects participate in activities and how they are affected by the activities.*
	Package Import	A package import is defined as a directed relationship that identifies a package whose members are to be imported by a namespace.
	Profile Application	A profile application is used to show which profiles have been applied to a package.*
	Protocol Transition	A protocol transition specifies a legal transition for an operation.*
	Realization	A realization is a specialized abstraction relationship between two sets of model elements, one representing a specification (the supplier) and the other represents an implementation of the latter (the client).*
	Component Realization	A component realization concept is specialized in the Components package to (optionally) define the Classifiers that realize the contract offered by a component in terms of its provided and required interfaces.*

APPENDIX: LIST OF ICONS





Icons of relationships

Icon	Title	Description
	Reply Message	A reply message is a reply message to an operation call.*
	Role Binding	A role binding is a mapping between features of the collaboration type and features of the classifier or operation. This mapping indicates which connectable element of the classifier or operation plays which role(s) in the collaboration.*
	Substitution	A substitution is a relationship between two classifiers which signifies that the substitutingClassifier complies with the contract specified by the contract classifier.*
	Template Binding	A template binding represents a relationship between a templateable element and a template.*
	Transition	A state transition usually has an event attached to it, but it is not necessary to attach one. If an event is attached to a state transition, the state transition will be performed when the event occurs.
	Usage	A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation.*

a) Material from the OMG UML Specification has been duplicated with permission.

Icons from Modules and Profile mechanism

Module, Profile, and Shared package model elements have their own icons, which are represented in the MagicDraw Browser. You may see the icons, their titles, and descriptions in the following table.

Icon	Title	Description
	Module	If you have (or developing) a large model, which has several weakly dependent parts, it is advisable to split it into several module files. Partitioning has a package level granularity.
	Profile	A Profile is a kind of Package that extends a reference metamodel. The primary extension construct is the Stereotype, which is defined as part of Profiles. ^a
	Profile Model Library exported as module	<p>This icon depicts a profile with applied Model Library stereotype.</p> <p>A package that contains model elements that are intended to be reused by other packages.*</p>
	Shared Package	<p>Not all module contents are visible in the working project. A module has a shared part and private part. Only the contents of the shared part are visible in the working project.</p> <p>The shared packages are marked with a hand image.</p>

a) Material from the OMG UML Specification has been duplicated with permission.

For more information about working with modules and profiling mechanism, see “Project Partitioning” on page 200.

INDEX

A

- About (command) 101, 103
- abstract
 - class 836, 857
 - operation 953
- Abstract (generalizable element) 326, 900
- action state 687
- activation bar 666, 925
- active
 - class 857
- activity diagram 685
- actor 660
 - working with 835
- aggregation 654
 - creating 844
- Align (command) 91
- assigning
 - classifier to classifier role 924
 - classifier to collaboration 864
 - classifier to instance 913
 - model element to a package 369
 - state to object flow state 947
- association 654
 - in class diagram 837
 - in use case diagram 837
 - n-ary 840
 - navigability of 844
 - with a role 852
- association class 839
- association end
 - multiplicity of 845
 - qualifier of 846
 - visibility of 845
- asynchronous message/stimulus 927
- attribute 849
 - creating new 849

- defining initial value of 913
- multiplicity 854
- scope of 853
- show only public 861
- type modifier of 853
- type of 852, 999

- attributes
 - controlling the list of 857
 - representing as association 852
 - sorting of 861
 - suppressing compartment 861

B

- binding dependency 882
- browser
 - changing position 109
 - changing size 109
 - closing or reopening 109
 - code engineering sets in 115
 - Containment tree 109
 - creating model elements and diagrams
 - in 124
 - Diagrams tree 118
 - displaying full information in 110
 - Documentation tab 126
 - functions of 106
 - Inheritance tree 120
 - Model Extensions tree 121
 - multiple selections 125
 - sorting alphabetically 109
 - structure 106
 - Zoom tab 127
- Browser (command) 99

C

- Call (event type) 1019
- Center (command) 91

- Change (event type) 1019
 - changeable 854
 - Check Syntax (command) 94
 - class 854
 - active 857
 - creating setters and getters 475
 - defining as abstract, leaf, root and/or
 - active 836, 857
 - description of 855
 - design patterns 859
 - inner elements of 856
 - owner of 369
 - show/hide package name 369
 - suppressing attributes 861
 - suppressing operations 861
 - class diagram 650
 - elements in 651
 - wizard 519
 - classifier
 - assign to an instance 913
 - assigning to a collaboration 864
 - assigning to classifier role 924
 - classifier role
 - assigning classifier to 924
 - in sequence diagram 666
 - Close All Diagrams (command) 100
 - Close All Projects (command) 81
 - closing
 - all opened projects 168
 - diagram 247
 - code engineering
 - checking syntax 116
 - creating new set 115
 - editing set 115
 - code generation 116
 - changing properties of 116
 - collaboration
 - assigning classifier to 864
 - collaboration diagram
 - changing numbering in 931
 - compartments of class 854
 - component 700, 866
 - diagram 698
 - instance 700
 - Component View package 167
 - composition 844
 - concurrency of operation 955
 - concurrent
 - operation 955
 - constraints
 - show-hide on symbol 343
 - showing on class 862
 - content diagram 752
 - Contents (command) 101, 102
 - Copy (command) 84
 - Copy as EMF (command) 85
 - Copy as JPG (command) 85
 - copying/cutting
 - among different projects 124
 - in the Browser 124
 - of text 279
 - symbol on a diagram 278
 - using drag and drop 277
 - CORBA IDL diagram 763, 776
 - creating
 - backup file 174
 - displaying already created relationships 556
 - elements and diagrams in Browser 124
 - new attribute 849
 - new code engineering set 115
 - new diagram 246
 - new operation 948
 - new project style 348
 - paths between shapes 263
 - creating:stereotypes 803
 - Cut (command) 84
- ## D
- data
 - creating several shapes of the same 259
 - Data package 167
 - data types 874
 - DDL diagram 765

- defining
 - model elements 325
 - Delete (command) 85
 - deleting
 - all model elements 125
 - from the Browser 124
 - symbol or model element 261
 - dependency
 - binding 882
 - permission 883
 - usage 883, 884
 - deployment diagram 699
 - design patterns 859
 - destroying sequence object 925
 - diagram
 - closing 247
 - creating 246
 - defining properties of 354
 - information table 257
 - opening 246
 - renaming 248
 - saving as image 315
 - diagrams
 - activity 685
 - class 650
 - collaboration 662
 - content 752
 - CORBA IDL 763, 776
 - DDL 765
 - implementation 698
 - robustness 757
 - sequence 665
 - state 674, 680
 - use case 659
 - web 759
 - WSDL 769
 - XML Schema 771, 775, 783
 - Diagrams menu 92
 - display
 - related elements 555
 - documentation of MagicDraw 37
 - drag and drop
 - copying 277
 - from browser to diagram 277
 - multiple symbols 277
 - source code files 278
 - drawing
 - more than one shape 258
 - shape 257
 - symbol from the Browser 124
- E**
- Edit menu 83
 - editing
 - code engineering set 115
 - editions of MagicDraw 23
 - Entertainment with UML (command) 101
 - enumeration 876, 987
 - enumeration literal 878
 - Environment (command) 93
 - event 675
 - Exit (command) 82
 - extend 887
 - extension point 1020
 - adding to use case 1022
- F**
- features of MagicDraw 23
 - File menu 80
 - Fit In Window (command) 87
- G**
- generalizable elements 900
 - defining as 900
 - generalization
 - grouping into tree 899
 - generate
 - code from the selected set 116
 - Generate Framework (command) 94
 - getter 475
 - grid 291
 - size 291
 - snapping to 291

- style 292
- visibility of 291
- Grid (command) 88
- grouping
 - diagrams in Browser 118
 - generalization paths into tree 899
 - of model elements 955
 - realizations into tree 995
- Guarded 955
- H**
- Help menu 101
- HTML text 375
- hyperlink 365
 - for model element 363
- hyperlink: for model element 363
- I**
- image
 - available formats 314
 - saving as 315
- implementation diagram 698
- Import Project (command) 81
- include 661, 906
- initial value
 - defining for attribute 913
 - showing on attribute 861
- inner elements
 - of class (inserting) 856
- installation
 - other platforms 51
 - system requirements 48
 - UNIX 51
 - using no install 51
 - Windows 50
- instance 704
 - assign classifier to 913
 - of component 700
 - of node 701
- interface 703
- interface style
 - making changes to 163

- multiple windows 164
- Interface Style (command) 93
- internal transition 922

J

- JVM (Java Virtual Machine)
 - for Solaris 50
 - for Windows 50

L

- layout 292
- Layout menu 89
- Leaf
 - class 857
- Leaf (generalizable element) 900
- lifeline 925
- link
 - in class diagram 657
 - in collaboration diagram 663
 - in implementation diagram 707
- link attribute 947
- link to self 663
- Look and Feel (command) 93
- Look and Feel themes 164

M

- Mac OS X
 - installation 51
- Magic Draw UML on the Web (command) 101
- MagicDraw
 - appearance of 163
 - documentation 37
 - editions 23
 - features 23
 - support 43
 - updating 67
- main toolbar 105
- Make Same Height (command) 91
- menus 80
- message
 - changing numbering of 931
 - defining action for 930

- diagonal 669
- in collaboration diagram 664, 665
 - asynchronous 927
 - synchronous 928
- in sequence diagram 935, 938
- predecessor of 934
- recursive 670
- to self 669
- uninterpreted 668
- Message Window (command) 100
- model
 - development of 22
 - static structural 650
- model element
 - defining 325
 - saving as image 315
- model management 650
- moving
 - of all model elements 125
 - symbol 277
- MS Office
 - copying to 278
- multiple windows 164
- multiplicity
 - of an attribute 854
 - of association end 845

N

- name
 - specifying for a shape 258
- n-ary association 840
- navigability
 - of association 844
- New Project (command) 80
- Next Diagram (command) 92
- node 700, 939
- note 797, 799
 - adding hyperlink to 365
 - retrieving documentation 801
 - text as HTML in 375
- note anchor 799, 800

- numbering
 - changing 931
- numbers
 - show/hide on collaboration diagram 930

O

- object flow state
 - assigning state 947
 - defining classifier for 947
- Open Project (command) 80
- opening
 - diagram 246
 - diagram from Browser 119
 - last project on startup 175
 - project 175
 - specification dialog box 325
- operation
 - concurrency 955
 - creating new 948
 - defining 948
 - defining as abstract, leaf, or root 954
 - functions of 948
 - parameter of 951
 - return type 953
 - return type of 950
 - show only public 861
 - showing signature of 860
 - visibility 954
- operations
 - controlling the list of 857
 - generation of setters and getters 857
 - managing 478
 - sorting of 861
 - suppressing compartment 861
- Options menu 93
- ordering
 - of attributes 861
 - of operations 861

P

- package 652, 702, 955
 - adding inner elements 956

- changing header position 957
- Component View 167
- Data 167
- dependency wizard 524, 528
- showing assigned model elements 957
- parameter
 - of an operation 951
- parameterized class 1015
 - defining of 1015
- partition 1010
- paste 278
 - with new data 278
- Paste (command) 84
- Paste With New Data (command) 84
- path 245
 - changing style of 264
 - defining properties of 354
 - drawing 263
- Paths (command) 85
- patterns See design patterns
- permission 883
- predecessor 934
- presentation options
 - for class 859
- Previous Diagram (command) 92
- primitive 879
- Print Active Diagram 82
- Print Options (command) 82
- Print Preview (command) 82
- printing 318
- project
 - dividing into modules 198
 - multiple projects 167
 - opening of 175
 - packages of 167
- Project (command) 93
- Projects (command) 82
- properties
 - changing code generation 116
- pseudostate 675

Q

- qualifier
 - of association end 846
- Quick Reverse (command) 94

R

- realization
 - grouping into tree 995
- Redo (command) 83
- Refresh (command) 87
- registering MagicDraw 67
- relationship
 - displaying on diagram 556
- renaming
 - diagram in Browser 119
- Retrieve DB Structure (command) 94
- return type
 - of an operation 953
- reverse
 - changed files 117
 - code engineering set 116
- robustness diagram 757
- Root
 - class 857

S

- Save (dialog box) 173
- Save As Image (dialog box) 315, 316, 317
- Save Project As (command) 80
- saving 173
 - as image 315
 - formats 173
 - in .mdf format 173
 - project 173
 - project as template 176
- Select All (command) 85
- selection 272
 - all symbols of the same type 272
 - multiple 272
 - multiple (in the Browser) 125
 - of all model elements 125

- of all symbols 272
- of symbol 272
- separator 801
 - adding hyperlink to 365
- sequence diagram
 - model elements in 666
 - overview 665
- Sequential 955
- setter 475
- shape
 - defining properties of 354
 - definition 245
 - drawing more than one shape 258
 - drawing of 257
- shortcut keys
 - assigning 165
- Signal (event type) 1019
- signal receipt
 - trigger event for 827, 831
- Solaris
 - JVM 50
- sorting
 - of attributes 861
 - of operations 861
- Space evenly (command) 91
- state 676, 681, 687, 1001
- state diagram
 - model elements in 675, 681
- state machine 675
- statechart diagram See state diagram
- stereotypes
 - defining properties of 354
 - show/hide on a symbol 343
 - showing on class 861
- stereotypes:creating of 803
- stimulus
 - changing numbering 930
 - changing numbering of 931
 - defining action for 930
 - main information about 664, 665
- Submit a Bug (command) 101, 103

- support for MagicDraw 43
- swimlane 1010
- symbols 245
 - defining properties 342
 - formatting 354
 - moving 277
 - presentation of
 - constraint 279
 - note 797, 279
 - separator 279
 - text box 279
- synchronous 928
- system boundary 661

T

- tagged values
 - show/hide on a symbol 343
 - showing on class 862
- template
 - saving as 176
- text box 279
 - adding hyperlink 365
 - text as HTML in 375
- text editor
 - choosing 117
- Time (event type) 1019
- Tip of the Day (command) 101, 102
- toolbars 102, 1026
- Tools menu 93
- transition 691
 - to self 680, 685
- type modifier 853
- type of an attribute
 - showing full path 861

U

- Undo (command) 83
- Unix
 - installation 51
- Unlock Key (command) 101
- updating MagicDraw 67

usage 883, 884

use case

- adding extension point 1022

- extension point 1020

V

View menu 87

visibility

- for operation 954

- of association end 845

- showing on attribute 861

- showing on operation 861

W

web diagram 759

Window menu 99

Windows

- JVM 50

wizards 387, 517

workspace 167

WSDL diagram 769

X

XMI 174

XML Schema diagram 771, 775, 783

Z

Zoom 1:1 (command) 87

Zoom In (command) 87

Zoom Out (command) 87

Zoom To Selection (command) 87

zooming 290

- adjusting step size 291

- fit in window 290

- to maximum size 290

- to original size 290

- using Browser 127

- zoom in 290

- zoom out 290